VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace adaptérů pro přístup k datům reprezentovaným časovými řadami

Implementation of Adapters for Access to Time Series Data

2016 Petr Zubek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Petr Zubek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Implementace adaptérů pro přístup k datům reprezentovaným časovými řadami
Implementation of Adapters for Access to Time Series Data

Jazyk vypracování: čeština

Zásady pro vypracování:

Diplomant se bude v rámci diplomové práce zabývat online datovými zdroji informací z různých služeb, mezi které patří například Google API, data poskytovaná společností INRIX a pod. Provede analýzu těchto služeb, jejich API, podmínek poskytování dat a popíše vlastnosti těchto služeb. V rámci implementační části práce se bude diplomant zabývat problematikou časových řad a implementací speciálních adaptérů, běžících na výpočetních uzlech HPC clusteru pro vybraný datový zdroj obsahující velký objem časových řad dostupný na IT4Innovations. V rámci implementací adaptéru bude brát ohled na zkušenosti získané analýzou API externích datových poskytovatelů. Dále pak bude vytvořen i adaptér pro tabulkový procesor MS Excel včetně porovnání obou přístupů k datům.

Jednotlivé body zadání jsou:
1. Analyzovat vybrané online datové zdroje se speciálními API na poskytování dat.
2. Analyzovat možnosti poskytování dat z datového zdroje definovaného vedoucím práce.
3. Implementace adaptéru pro čtení dat pro výpočty na HPC clusteru.
4. Implementace adaptéru pro čtení dat v rámci tabulkového procesoru MS Excel.
5. Příprava dat pro experimenty a provedení experimentů.

Seznam doporučené odborné literatury:

[1] Google API Documentation: https://developers.google.com/products/
[2] INRIX Dev Zone: http://inrix.com/resources/dev-zone/
[3] Intel: Improving traffic management with big data analytics,
http://www.intel.com.br/content/dam/www/public/us/en/documents/case-studies/big-data-xeon-e5-

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Kateřina Slaninová, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka	prof. RNDr. Václav Snášel, CSc.
vedoucí katedry	*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Rád bych upřímně poděkoval Ing. Janu Martinovičovi, Ph.D. za podnět ke vzniku práce, jeho náměty, připomínky a cenné rady. Dále bych chtěl poděkovat vedoucí mé diplomové práce Ing. Kateřině Slaninové, Ph.D. za její praktické připomínky, rady a postřehy. Mé díky také patří Bc. Martinovi Moudrému za poskytnutí výchozích podkladů k tvorbě této práce.

Abstrakt

Tato práce se zabývá analýzou dostupnosti dopravních otevřených dat, komerčních dopravních dat a analýzou dat poskytovaných Google API. Poté se zabývá jejich analýzou, definováním vlastností a podmínek poskytování nalezených zdrojů dat. Dále práce řeší popis implementace adaptérů pro formát HDF5, specifikaci problematiky při implementaci a definici vzniklých prvků architektury, které jsou uvedeny v jednotlivých kapitolách. V práci je popsána funkcionalita generátoru dat se zkušenostmi získanými analýzou dopravních dat. Následně jsou generovaná data využita pro provedení experimentů nad vytvořenými adaptéry pro formát HDF5 na výpočetních uzlech HPC.

Klíčová slova: OpenData, Google API, adaptér, Excel doplněk, HDF5, HPC

Abstract

This thesis analyzes the availability of transport open data, commercial transport data and analysis of data provided by the Google API. It then discusses their analysis, defining the characteristics and conditions of the provision of found data sources. The work also addresses description of the implementation adapters HDF5 format, specification of problems in implementation and a description of the architecture elements that are described in each chapter. The thesis describes the functionality of a generator of data with respect the experience analyzing traffic data. Subsequently, the generated data used to perform experiments for created adapters HDF5 format on HPC compute nodes.

Key Words: OpenData, Google API, adapter, Excel add-in, HDF5, HPC

Obsah

Seznam použitých zkratek a symbolů

ABI	–	Application binary interface
API	–	Application Programming Interface
BTS	–	Base Transceiver Station
CC	–	Creative Commons
CFCD	–	Cellular Floating Car Data
CLI	–	Common Language Infrastructure
CSS	–	Cascading Style Sheets
CSV	–	Comma-Separated Values
DAB	–	Digital Audio Broadcasting
DLL	–	Dynamic-link library
DMB	–	Digital Multimedia Broadcasting
ECMA	–	European Computer Manufacturers Association
FCD	–	Floating Car Data
FM	–	Frequency Modulation
GCC	–	GNU Compiler Collection
GFCD	–	GPS Floating Car Data
GNSS	–	Global Navigation Satellite System
GNU FDL	–	GNU Free Documentation License
GNU	–	GNU's Not Unix!
GPRS	–	General Packet Radio Service
GPS	–	Global Positioning System
GSM	–	Global System for Mobile telecomunication
HDF5	–	Hierarchical Data Format 5
HTML	–	HyperText Markup Language
HTTP	–	Hypertext Transfer Protocol
ID	–	Identifikátor
IIS	–	Internet Information Services
IO	–	Input/Output
IOPS	–	Input/output operations per second
ISO	–	Application Programming Interface
ITS	–	Intelligent Transportation System
JSON	–	JavaScript Object Notation
LED	–	Light-Emitting Diode
ODbL	–	Open Database License
OpenMP	–	Open Multi-Processing
PDDL	–	Public Domain Dedication and Licence

PDF	–	Portable Document Format
PHP	–	Hypertext Preprocessor
PInvoke	–	Platform Invocation Services
PInvoke	–	Platform Invocation Services
PST	–	Pacific Standard Time
RDF	–	Resource Description Framework
RDS-TMC	–	Radio Data System - Traffic Message Channel
REST	–	Representational State Transfer
RFC	–	Request For Comments
SaaS	–	Software as a Service
SMS	–	Short Message Service
SOAP	–	Simple Object Access Protocol
SSD	–	Solid-state drive
TEC	–	Traffic event compact application
TFP	–	Traffic flow and prediction application
TPEG	–	Transport Protocol Experts Group
URI	–	Unified Resource Identifier
URL	–	Uniform Resource Locator
UTC	–	Coordinated Universal Time
VBA	–	Visual Basic for Applications
VSTO	–	Visual Studio Tools for Office
W3C	–	World Wide Web Consortium
WCF	–	Windows Communication Foundation
WEA	–	Weather information application
WP	–	Worker proces
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V dnešním světě plném moderních technologií vzniká potřeba neustále zdokonalovat technologie, aby byly stále aktuální. Abychom dosáhli moderních technologií, je potřeba měřit a analyzovat informace, chceme-li data. Jedním z cílů je mít inteligentní dopravní systémy, které se budou zdokonalovat. K dosažení tohoto vysokého cíle vzniká potřeba měřit, následně získat dopravní data, abychom je mohli analyzovat. Dopravní data jako taková jsou cennými informacemi a dle toho jsou náležitě zpoplatněna. Proto se tato práce zabývá nalezením vhodného zdroje dopravních dat, ideálně otevřených dopravních dat. A následně jejich ukládáním a reprezentací.

V počáteční kapitole se zabýváme vysvětlením, co to jsou OpenData, dále si rozebereme Google API, kde prozkoumáme vybrané části API, budeme hledat spojitosti s dopravními daty, popíšeme si jak přistupovat do Google API. Popíšeme si implementaci služby pro Google API a licenční podmínky Google API. Objasníme si problematiku dopravních OpenDat, stručně si popíšeme metody, jak jsou dopravní data získávány a budeme analyzovat a popisovat vlastnosti nalezených otevřených zdrojů. V následující kapitole se budeme zaobírat nalezenými komerčními datovými zdroji, jako jsou např. INRIX, TomTom a Here. Tyto nalezené zdroje budeme taktéž analyzovat, studovat jejich vlastnosti a objasníme si jejich licenční politiku. Další kapitola popisuje implementaci adaptérů pro formát HDF5, kde si popíšeme detailně vzniklé varianty adaptérů. Následně si popíšeme problematiku, která nastala při vývoji adaptérů. Objasníme si vzniklou architekturu a její jednotlivé prvky, kterými jsou webová služba, klientská knihovna webové služby, multiplatformní verze adaptéru a v poslední řadě si popíšeme naimplementovaný doplněk pro Excel, kterému se věnuje další kapitola. Ohledně implementace doplňku pro Excel si analyzujeme možnosti vývoje doplňků pro Excel. Poslední kapitola se věnuje experimentům provedenými nad jednotlivými prvky architektury. Při výkonu experimentů je využito i výpočetních uzlů superpočítače (HPC).

2 Analýza vybraných online datových zdrojů se speciálními API na poskytování dat

V této kapitole si objasníme, co jsou OpenData a jaké je jejich použití. Následně si popíšeme vlastnosti a nabídku Google API, jak ke Google API přistupovat a využívat tyto služby.
Popíšeme si licenční podmínky z pohledu programátora. Dále budeme analyzovat dostupnost dopravních OpenDat. Popíšeme si metody, jak jsou tato data sbírána. Analyzujeme vlastnosti dopravních OpenDat, které jsme nalezli.

2.1 OpenData

V dnešním počítačovém světě nalezneme nepřeberné množství různých, ať už online nebo offline, datových zdrojů. Tyto datové zdroje velmi napomáhají zlepšit produktivitu v důležitých odvětvích, jako je např. průmysl, zemědělství, energetika a služby, ale mohou být i přínosem pro vědu a výzkum. Jako příklad online datového zdroje lze uvézt kurzovní lístek na internetu, který může být jednoduše zpracován nějakým strojem. Proto můžeme chápat jako online datový zdroj taková data, která jsou dostupná na internetu a jsou ve strojově čitelném formátu [1]. Nejčastěji jsou online datovým zdrojem různá webová API, ale mohou to také být různé specifické databáze, které jsou dostupné na internetu apod. Online datový zdroj je mnohdy dostupný na internetu za pomocí webové služby, REST, CSV, XML či soubor ke stažení v nějaké binární podobě.

Nemohu opomenout pojem Open data (otevřená data), který nabývá na své popularitě [2]. Jako jedním z online datových zdrojů můžeme považovat právě Open data. Principem Open dat je, jak už vyplývá z názvu, aby data byla otevřená, tedy dostupná všem s možností data dále publikovat. Open data jsou nejvíce publikována pod licencí Creative Commons (CC,CC0), Open Data Commons Public Domain Dedication and Licence (PDDL) a Open Data Commons Open Database License (ODbL) a jiné. Toto jsou doporučené licenční modely pro účely Open dat. Ovšem lze používat jiné licenční modely, ale ty nejsou doporučovány jako např. GNU Free Documentation License (GNU FDL), Against DRM atd. U licence CC je třeba si dávat pozor na její podtřídy, ne všechny podtřídy jsou vhodné pro Open data. Creative Commons má například tyto podtřídy: Creative Commons CCZero (CC0), Creative Commons Attribution 4.0 (CC-BY-4.0), Creative Commons Attribution Share-Alike 4.0 (CC-BY-SA-4.0) atd. Pro příklad pár nevhodných: Creative Commons No-Derivatives Licenses, Creative Commons Developing Nations License apod. Typy licencí jsou od sebe rozdílné a slouží pro různá data a různé účely. Ovšem cílem práce není detailně zkoumat licenční politiku Open dat. Základní znaky Open dat jsou následující, data jsou úplná, snadno dostupná, strojově čitelná, používající standardy s volně dostupnou specifikací, zpřístupněna za jasně definovaných podmínek užití dat s minimem omezení, dostupná uživatelům při vynaložení minima možných nákladů. Nemohu opomenout pojem Open data (otevřená data), který nabývá na své popularitě. Jako jedním z online datových

zdrojů můžeme považovat právě Open data. Principem Open dat je, jak už vyplývá z názvu, aby data byla otevřená, tedy dostupná všem s možností data dále publikovat [2]. Open data jsou nejvíce publikována pod licencí Creative Commons (CC,CC0), Open Data Commons Public Domain Dedication and Licence (PDDL) a Open Data Commons Open Database License (ODbL) a jiné. Toto jsou doporučené licenční modely pro účely Open dat. Ovšem lze používat jiné licenční modely, ale ty nejsou doporučovány jako např. GNU Free Documentation License (GNU FDL), Against DRM atd. U licence CC je třeba si dávat pozor na její podtřídy, ne všechny podtřídy jsou vhodné pro Open data. Creative Commons má například tyto podtřídy: Creative Commons CCZero (CC0), Creative Commons Attribution 4.0 (CC-BY-4.0), Creative Commons Attribution Share-Alike 4.0 (CC-BY-SA-4.0) atd. Pro příklad pár nevhodných: Creative Commons No-Derivatives Licenses, Creative Commons Developing Nations License apod. Typy licencí jsou od sebe rozdílné a slouží pro různá data a různé účely. Ovšem cílem práce není detailně zkoumat licenční politiku Open dat. Základní znaky Open dat jsou následující, data jsou úplná, snadno dostupná, strojově čitelná, používající standardy s volně dostupnou specifikací, zpřístupněna za jasně definovaných podmínek užití dat s minimem omezení, dostupná uživatelům při vynaložení minima možných nákladů [3].

Open data mají požadavky na jejich užívání: „neomezují jejich uživatele ve způsobu použití dat, opravňují uživatele k jejich dalšímu šíření, musí být uveden autor dat (i při dalším šíření), při dalším šíření musí i ostatní uživatelé mít stejná oprávnění s daty nakládat - během šíření dat nesmí dojít např. k omezení jejich využití pouze pro nekomerční účely."[3]

Open data mají definován stupeň otevřenosti [4], chceme-li stupeň kvality otevřenosti, kdy 1 je nejnižší a 5 nejvyšší. O tvorbu tohoto ohodnocení se zasloužil informatik, fyzik, programátor a vysokoškolský pedagog Timothy John Berners-Lee, tvůrce World Wide Webu a ředitel konsorcia W3C. Jednotlivé stupně otevřenosti jsou uvedeny v příloze A.

2.2 Google API

Společnost Google nabízí své API ke svým produktům, je jich nepřeberné množství [5]. Například API pro Ad Exchange (trh pro prodej a nákup reklamy v reálném čase), Ad Sense (systém pro zobrazování relevantních reklam), Blogger, Books, Calendar, Drive, Contacts, Maps, Gmail, Google Cloud, YouTube a spousta dalších API. Pro příklad si popíšeme tyto následující API:

- Drive API – poskytuje API pro přístup a práci s Google Drive, což je diskové úložiště. Zde bylo cílem synchronizovat soubory mezi Google Drive a informačním systémem.
- Contacts API – poskytuje API pro přístup ke Google kontaktům. Bylo potřeba poskytnout rozhraní pro synchronizaci kontaktů mezi Google a informačním systémem.
- Calendar API – poskytuje API pro přístup ke Google kalendáři. Zde bylo třeba taktéž poskytnout rozhraní, které umožňovalo synchronizovat události v kalendáři Google a kalendáři informačního systému.

- Maps API – poskytuje API pro práci a přístup k mapovým podkladům společnosti Google. Bylo nutno generovat URL adresu, na které byla dostupná statická mapa (mapa jako obrázek) včetně nadefinovaných připínáčků na mapě. Dále bylo nutné poskytnout mapu, která byla dynamická (JavaScript kód) včetně připínáčku.

Pro využívání Google API, je třeba si na stránkách pro Google vývojáře [1] vytvořit tzv. projekt, ten je potřeba pro získání API účtu a identifikátoru pro vyvíjenou aplikaci. Při vytváření je nutno si zvolit jméno projektu a popřípadě zvolit datacentrum pro komunikaci (pro evropský nebo americký trh). Po úspěšném vytvoření projektu je přejdeme na stránku správce API (API Manager), kde je třeba povolit Google API, které chceme používat. Pro ilustraci si můžeme povolit Drive API, Contacts API a Calendar API. Pro Maps API není třeba povolení, protože lze přistupovat skrze veřejné API. Dále je nutno si vytvořit přístupové údaje (Credentials), kde jedním z požadavků je vytvoření OAuth klientské ID. Dále je třeba nastavit upravení obrazovky, kde souhlasí klient s přístupem pomocí OAuth (OAuth consent screen), zde je potřeba pouze nastavit název aplikace. Poté je potřeba vybrat typ aplikace. Pro příklad zvolíme webovou aplikaci, kde je nutno nastavit autorizační URI adresu, na kterou bude uživatel přesměrován v případě správného přihlášení. Pro OAuth přístup je potřeba obdržet přístupové údaje, které obsahují klientské ID a heslo klienta. V rámci API účtu můžeme sledovat počet požadavků a aktuální využití kvóty.

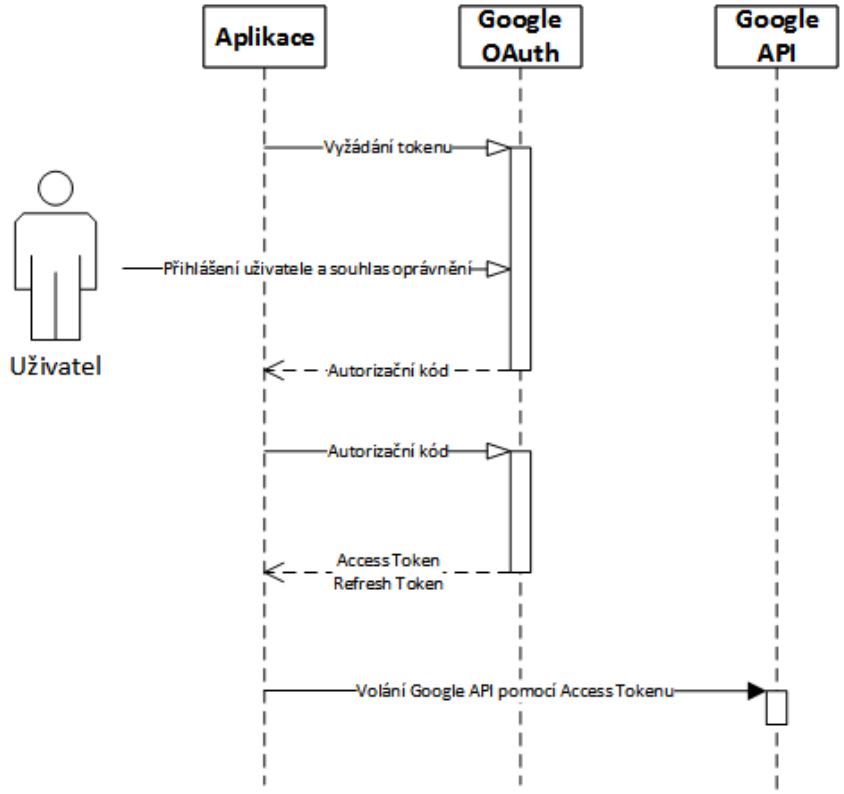
Popíšeme si OAuth autentizaci. OAuth autentizace se řeší, pokud dané API není veřejně dostupné. Je li třeba se autentizovat u Google API, tak nejčastější způsob je pomocí otevřeného protokolu OAuth. Tento protokol si klade za cíl autentizovat a autorizovat uživatele pomocí webů třetích stran (Facebook, Microsoft, Google, Twitter atd.). Výhodou je, že služba, která poskytuje přihlášení pomocí OAuth, nemusí ukládat hesla ve své databázi (heslo je uloženo u třetí strany). Celý proces přihlášení pomocí OAuth je znázorněn pomocí sekvenčního diagramu na obrázku 1.

Aby se uživatel mohl úspěšně přihlásit, je třeba získat od OAuth poskytovatele následující údaje, ID klienta a klientské heslo. Poskytnuté údaje mohou vypadat následovně, viz obrázek 2.

V případě Googlu, je třeba specifikovat autorizační URI, tuto jsme si specifikovali ve fázi vytvoření API účtu. Pokud se jedná o první přihlášení, je třeba schválit oprávnění pro aplikaci k přístupu k datům, která má uživatel uložena na Googlu. Oprávnění je definováno pomocí rozsahů oprávnění např. zapisovat na Google Disk, číst Google Disk apod.

Při první komunikaci se musí vyžádat tzv. Refresh Token, tedy aktualizační token, který slouží k tomu, abychom si mohli později vyžádat tzv. Access Token, tedy přístupový token. Aktualizační token je třeba si někam uložit, např. do databáze. Přístupový token slouží k jednotlivým operacím s Google API, tedy lze chápat, že jeden dotaz do Google API vyžaduje jeden přístupový token. Problém nastává ve chvíli, kdy aplikace, která využívá Google API má fungovat jako služba. Google nabádá k využívání metody „GoogleWebAuthorizationBro-

[1] adresa webu pro vývojáře https://console.developers.google.com



Obrázek 1: Sekvenční diagram OAuth komunikace

Client ID for Web application

Client ID	1015625894217-0u7jp59n9loee61nvc8rokub4j2kl61t.apps.googleusercontent.com
Client secret	Vys62iZxzoHozKah9Ufl_Oto

Obrázek 2: OAuth přihlašovací údaje

ker.AuthorizeAsync(secrets, scopes, userName, CancellationToken.None);“, ovšem tato metoda má jeden háček. Knihovna Google Apis OAuth2 pro .NET Framework je naprogramována tak, aby se v momentě, kdy je třeba se přihlásit či schválit oprávnění, otevřel prohlížeč se stránkou pro přihlášení ke Googlu. Tento způsob je nevhodný, neboť žádný uživatel nebude mít přístup k serveru, na kterém je provozována služba pro přístup do Google API. Ovšem existuje řešení, jak tento problém obejít.

Nejprve je nutné uživateli „předhodit“ URL adresu, pomocí které se přihlásí ke svému Google účtu a poskytne aplikaci patřičná oprávnění. Jak se adresa generuje, můžeme vidět na následujícím fragmentu kódu 1. (Klientské ID můžeme chápat jako ID aplikace).

```
public String RequestPermissionURL(String clientID)
{
    //...
    String googleAuth = "https://accounts.google.com/o/oauth2/auth";
    String access_type = "access_type=offline";
    String response_type = "response_type=code";
    String client_id = "client_id=" + clientID;
    String authURL = "redirect_uri=" + redirect_uri; //autorizacni URI
    String scope = "scope="; //rozsah opravneni

    AddScopesToPermissionAll();

    for (int i = 0; i < this.scope.Count; i++)
    {
        scope += this.scope[i] + " ";
    }

    String URL = googleAuth + "?" + access_type + "&" + response_type + "&" +
        client_id + "&" + authURL + "&" + scope;
    //...
}
```

Výpis 1: Metoda pro generování autorizační adresy

V momentě, kdy uživatel schválil oprávnění, získá autorizační kód, se kterým je přesměrován na stránku, která je specifikována jako autorizační URI. Níže můžeme vidět metodu 2, na kterou je přesměrován uživatel po úspěšném přihlášení a schválení práv.

```
[WebGet(UriTemplate = "?code={code}")]
public string Authorize(string code)
{
    Customer cust = new CustomerTable().SelectByIncomeToken();
    ClientSecrets secrets = new ClientSecrets { ClientId = cust.clientID,
        ClientSecret = cust.clientSecret };
    var init = new GoogleAuthorizationCodeFlow.Initializer() { ClientSecrets =
        secrets, Scopes = new Auth().GetScopesForAuth() };
    IAuthorizationCodeFlow flow = new GoogleAuthorizationCodeFlow(init);
    TokenResponse tr = flow.ExchangeCodeForTokenAsync(secrets.ClientId, code,
        ConfigurationManager.AppSettings["AuthorizeURL"], CancellationToken.None
        ).Result;
    cust.incomeToken = false;
    cust.refreshToken = tr.RefreshToken;
    cust.accessToken = tr.AccessToken;
    //...
}
```

Výpis 2: Metoda pro zpracování autorizace

Google vrátí uživatele na autorizační URI s parametrem code, který obsahuje autorizační kód pro získání aktualizačního a přístupového tokenu. Z databáze se získá záznam uživatele, vytvoří objekt ClientSecrets, který obsahuje klientské ID a klientské heslo. V metodě Initializer k objektu ClientSerets je definován rozsah oprávnění. Vytvoří se objekt IAuthorizationCodeFlow, pomocí kterého lze volat token. Metoda ExchangeCodeForTokenAsync vyžaduje klientské ID, autorizační kód, autorizační URI a token pro zrušení operace. Poté je získán objekt TokenResponse, který obsahuje aktualizační token a přístupový token, který je využit pro první operaci. Pro další přístup už dostačuje pouze aktualizační token, který je svázán s klientským ID a účtem uživatele. Pro příští přihlášení stačí pouze volat následující fragment kódu 3.

```
//...
var init = new GoogleAuthorizationCodeFlow.Initializer() { ClientSecrets =
    secrets, Scopes = scopes };
IAuthorizationCodeFlow flow = new GoogleAuthorizationCodeFlow(init);

TokenResponse tr = flow.RefreshTokenAsync(secrets.ClientId, cust.refreshToken,
    CancellationToken.None).Result;
credential = new UserCredential(flow, secrets.ClientId, tr);
//...
```

Výpis 3: Metoda pro přihlášení uživatele

V objektu TokenResponse je opět přístupový token. Toto je postup, jak získat přístup do Google API pomocí standardních knihoven Google Apis pro .NET Framework, aniž by byl otevřen prohlížeč na straně služby, která je provozována na serveru.

Jak bylo zmíněno na začátku, služba byla vytvořena za pomocí .NET Frameworku. Google ke svým API nabízí připravené knihovny, pomocí kterých lze snadno volat jednotlivé metody, jež vrací odpověď jako objekt daného programovacího jazyka. Google nabízí knihovny pro celou řadu programovacích jazyků, ale i skriptovacích jazyků, jako je např. Java, JavaScript, .NET, Objective-C, PHP, Python apod. Klientské knihovny jsou nejčastěji adapterem (wrapperem) Google REST API.

Knihovna pro Google **Drive API** nabízí standardní volání API. Tedy vytvoření, nahrání, stažení, aktualizaci a smazání souboru. Vytvoření, smazání složek a získání obsahu složek. Úprava oprávnění k souborům, získání odkazů ke sdílení. V neposlední řadě aktualizace metadat souborů a práce s košem. Při práci s Google Drive API byla shledána velkým nedostatkem absence metody, k získání adresářové struktury včetně složek a souborů a jejich cesty z kořenového adresáře nebo konkrétního adresáře. Je proto nutné si takovou metodu naimplementovat tak, že rekurzivně prochází adresářovou strukturu a sestavuje seznam absolutní cesty k souborům a složkám.

Další knihovna Google **Contacts API** poskytuje přístup ke Google kontaktům a umožňuje vytvářet, číst, upravovat a mazat kontakty. Zde mapuje objekty mezi daným systémem a Google kontakty. Dále se ověří a konzultuje, které údaje se mají mapovat a které nikoliv. Nyní se řeší synchronizace jména včetně titulů, přezdívek, webových stránek, emailů, telefonů, firem, poznámek a adres.

Knihovna Google **Calendar API** nabízí přístup ke Google kalendáři, kde API umožňuje měnit přístupová práva k události, vytvořit, číst, upravovat a mazat události. Je potřeba naimplementovat metodu, která získá více událostí v uživatelem definovaném rozsahu. Pomocí této metody by daný informační systém porovnával svůj kalendář. Jsou synchronizovány tyto atributy: datum vytvoření, popis, počátek události, konec události, počáteční časová zóna, koncová zóna, lokalita, způsoby a čas upozornění, nadpis události, čas aktualizace události, zda je to celodenní událost, identifikátor opakované události a popřípadě nastavení opakované události. Samozřejmě Google Calendar API nabízí ještě více atributů, ale ty nejsou nutné pro synchronizaci. Mapování opakovaných událostí je trochu složitější. Interně v Google kalendáři vytvoření opakující se události funguje následovně, vytvoří se tzv. rodičovská událost, která má vyplněn atribut „recurrence“, dle RFC 5545 [2] [6]. Poté jsou vytvořeni potomci, to jsou takové události, které mají vyplněn atribut „recurringEventId“, které se odkazují na rodičovskou událost. Ovšem Google neimplementuje veškerou funkcionalitu, která je popsána v RFC 5545, ale jenom její část. Popíšeme si funkcionalitu implementovanou Googlem. Atribut „recurrence“ většinou začíná klíčovým slovem „RRULE“, čímž říkáme, že se jedná o opakující se pravidlo. Dále je klíčové slovo „FREQ“, toto klíčové slovo udává frekvenci událostí, může nabývat hodnot „SECONDLY“ (po

[2] RFC jsou dokumenty se sadou doporučení pro technologie, protokoly apod.

vteřině), „MINUTELY“ (po minutě), „HOURLY“ (po hodině), „DAILY“ (denně), „WEEKLY“ (týdně), „MONTHLY“ (měsíčně), „YEARLY“ (ročně). Dále můžeme specifikovat klíčové slovo „INTERVAL“, které obsahuje celočíselnou hodnotu, která značí interval opakování. Mějme příklad, kdy frekvence je měsíčně a interval opakování je 2, znamená to tedy, že se jedná o událost, která se bude opakovat co 2 měsíce; pro hodnotu 3 co 3 měsíce atd. Pro frekvenci denně s hodnotou intervalu opakování 6, se bude opakovat co 6 dní. Dalším z klíčových slov je klíčové slovo „BYDAY“. Toto klíčové slovo udává den v týdnu a může být kombinováno s celým číslem. Hodnoty jsou „SU“, „MO“, „TU“, „WE“, „TH“, „FR“, „SA“. Číslo je napsáno před hodnotou a značí pořadí. Například „4MO“ znamená každé 4. pondělí v měsíci. Následující klíčové slovo je „COUNT“, jež nám říká, kolikrát je událost opakována. Posledním klíčovým slovem je „UNTIL“, které musí být ve formátu splňujícím RFC 3339 [33] (jedná se o doporučení k formátu, který reprezentuje časový údaj). Toto klíčové slovo specifikuje, do jakého data se má událost opakovat. Všechna klíčová slova lze mezi sebou kombinovat a některá nemusí být uvedena. Pro opakované události se vytvoří objekt, který reprezentuje jednotlivá klíčová slova pro mapování a je nutno tato klíčová slova „parsovat“ a skládat je zpět do validního formátu dle RFC 5545.

Google pro své **Maps API** neposkytuje knihovnu, ale API lze volat skrze REST API. REST je architektonický styl, který definuje sadu architektonických principů, kterými lze navrhnout webové služby, které se zaměřují na zdroje systému (data) [7]. Zohledňuje také způsob, jakým jsou stavy zdrojů adresovány a přenášeny skrze HTTP širokou škálou klientů, kteří mohou být napsáni v různých programovacích jazycích. Mezi základní principy návrhu patří: používat HTTP metody explicitně, být bezstavový, vystavovat adresářovou strukturu jako URI, podpora různých reprezentací (XML, JSON apod.). Ovšem existují komunitní implementace Google Maps API knihoven pro .NET, Javu a Go. Tyto knihovny usnadní práci s API a proto není třeba řešit komunikaci s API (volání REST API), zpracování reprezentace (JSON, XML apod.) a další problematiku ohledně Maps API. Maps API se rozpadá do několika dalších API [8]. Google Maps Geocoding API řeší konverzi mezi adresou a geografickými souřadnicemi, Google Places API Web Service řeší problematiku automatického doplňování a získávání aktualizovaných informací ohledně objektů, budov v žádané lokalitě, The Google Maps Elevation API nám získává informace o nadmořské výšce, Google Maps Roads API poskytuje informace o silnicích a jejich metadata (maximální povolená rychlost na úseku apod.), Google Maps Geolocation API čerpá data z buňkových sítí a WiFi, které může mobilní klient detekovat a na základě těchto dat nám poskytne lokaci, Google Maps Directions API poskytuje informace k zadaným bodům ve formě itineráře (po 500m odboč vlevo, pak jeď rovně atd.), Google Maps Time Zone API poskytuje informace o časových zónách k dané lokaci a poslední Google Maps Distance Matrix API počítá délku a dobu trvání trasy mezi danými body. Tyto všechny API, tedy Maps API, zabaluje do sebe komunitní knihovna „Google Maps Web Services wrapper API for .NET“ [3], dále jen komunitní knihovna. Google Maps API jako jediné nevyžaduje API klíč a to i přesto, že je to uvedeno v dokumentaci. Vytvoří se metoda, která vrací URL adresu, na které je vygene-

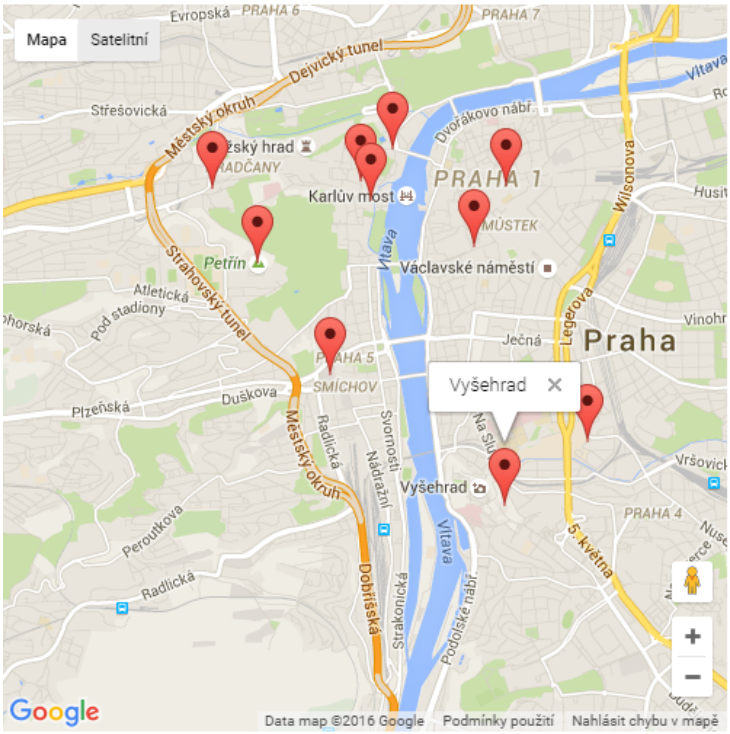
[3] adresa webu komunitní knihovny https://github.com/maximn/google-maps

Obrázek 3: Náhled na vygenerovanou statickou mapu

rovaná statická mapa (obrázek) s připínáčky zadaných lokací. Klíčovou metodou pro registraci připínáčku jsou parametry ID klienta, adresa, kde má být umístěn připínáček a poslední parametr nadpis připínáčku. Tato metoda volá interní metodu pro získání pozice („getPosition“), kde je vstupem adresa a výstupem GPS souřadnice a tato interní metoda využívá funkcionalit komunitní knihovny, konkrétně části API Google Maps Geocoding API. Poté se GPS souřadnice uloží do databáze k patřičnému ID klienta. Pro vygenerování statické mapy, která má parametry „mapCenter“, což je adresa, která se převede na GPS souřadnice a dle těchto souřadnic se mapa vycentruje, dalším z parametrů je šířka a výška mapy v pixelech, velikost přiblížení mapy a poslední parametr ID klienta. Pak se načtou všechny GPS lokace připínáčků z databáze patřičného klienta a zavolá se metoda komunitní knihovny, pomocí které získám validní URL adresu, na které je dostupná statická mapa na obrázku 3.

Bohužel statická mapa má nevýhodu, že není tolik interaktivní jako dynamická mapa. S dynamickou mapou, můžeme přibližovat a oddálit mapu na webové stránce, kliknout na připínáček a dozvědět se informace o připínáčku, připínáček obsahuje animaci při načtení stránky. U metody pro dynamickou mapu jsou parametry stejné, jako u statické mapy. Rozdíl je v tom, že se nevrací URL adresa, ale minifikovaný JavaScript kód, který se vloží do HTML stránky. Vykreslený výsledek HTML stránky s patřičným kódem je na následujícím obrázku 4.

Upřesníme si, co znamená minifikace. Minifikace je optimalizační proces, který má za cíl zmenšit velikost zdrojového kódu, ať už JavaScript, CSS nebo nějaká jiná technologie obsahující zdrojový kód, a zachovat funkcionalitu kódu [9]. Obecně má smysl ji používat s webovými



Obrázek 4: Náhled na vygenerovanou dynamickou mapu

technologiemi, kdy klient stahuje data (kód) ze serveru a jedná se o větší rozsah kódů.

Většina poskytovaných Google API jsou omezena kvótou požadavků. Jedno volání API se rovná jednomu požadavku. Každé API má jinak velikou kvótu. Kvóta bývá určena na den, ale není to pravidlem. Zároveň může být kvóta na nějaký krátký časový úsek a ještě aplikována na uživatele, např. 1000 požadavků na 100 sekund na uživatele. V případě Drive API je kvóta 1 miliarda požadavků na den a 1000 požadavků na 100 sekund na uživatele. Je dobré poznamenat, že kvóta se resetuje o půlnoci Tichomořského standardního času (PST, UTC -8:00). U Calendar API je kvóta 1 milión požadavků na den a 500 požadavků na 100 sekund na uživatele. Contacts API má nastavenu kvótu 20 miliónů požadavků na den a 1000 požadavků na 100 sekund na uživatele. Google své API poskytuje zdarma, samozřejmě se může stát, že by nám počet požadavků nestačil. Poté je třeba zvolit si plán, který již bude zpoplatněn [10]. Google poskytuje dva základní plány, plán Standard a plán Premium. Plán Standard je navržen tak, že zvyšuje kvótu na vyšší hodnoty a aktivaci tohoto plánu lze provést pohodlně přímo z internetu, pokud přečerpáme limit, pak se dokupuje balíček požadavků za určitou cenu dle API. V případě plánu Premium je třeba kontaktovat zástupce Google, kde si klient sestaví plán na míru. Pokud Google veřejně nenabízí plány, tak na portálu Google Developers v rámci API účtu lze navýšit kvótu pomocí formuláře, kde je nám sestaven plán na míru dle plánu Premium.

Co se týče licenčních podmínek, jsou stanoveny obecné licenční podmínky pro celé Google API a mohou být definovány dodatečné licenční podmínky pro konkrétní API [11]. Zde si uvedeme licenční podmínky ve zkratce (jedná se o výtah toho nejpodstatnějšího z hlediska vývojáře).

Pokud by nastal rozpor mezi obecnými licenčními podmínkami a dodatečnými licenčními podmínkami, pak mají vždy přednost dodatečné licenční podmínky. Jedním z licenčních požadavků je, že k API se bude přistupovat pomocí prostředků API, jak jsou popsané v dokumentaci. Pokud nám Google přidělí pověření vývojáře (klientské ID), je nutné jej používat s platnými API. Neměli bychom žádným způsobem zkreslovat či maskovat při používání API nebo účtu vývojáře svou identitu nebo identitu API klienta. Dále se zavazujeme, že nebudeme obcházet kvóty jednotlivých API. Pokud budeme potřebovat navýšit kvótu, kontaktujeme příslušný tým Google API pro informace. Google monitoruje svá API pro zlepšení služeb a produktů a neměli bychom zasahovat do tohoto sledování. Google může pozastavit přístup k API bez výpovědní lhůty, pokud dojde k domnění, že jsme v rozporu s podmínkami. Měli bychom věnovat rozumné úsilí k tomu, abychom ochránili uživatelské informace, včetně osobních údajů. Pokud by došlo k neautorizovanému přístupu nebo užívání, měli bychom neprodleně uvědomit uživatele o této události. Je zakázáno: přeprodávat API třetím stranám, zavádět do produktů a služeb společnosti Google viry, červy, vady, Trojské koně, malware nebo jakékoliv položky destruktivní povahy, urážet uživatele a zneužívat jeho data, či je nějak pozměňovat, použití API v nějakém případě, kdy selhání API může vést ke smrti, osobnímu zranění nebo poškození životního prostředí (jako je provoz jaderných zařízení, řízení letového provozu nebo podpůrné systémy života).

Hlavním cílem této práce je zaměření na dopravní časové řady, ale Google prozatím neposkytuje API dopravních dat v reálném čase. Avšak poskytuje informace o dopravě ve svých mapových podkladech.

2.3 Dopravní OpenData

Cílem této části je nalézt vhodná dopravní OpenData. Získat taková data je nelehký úkol, ač je neustálá iniciativa ze strany Evropské unie, aby členské země tato data pokud možno zveřejňovaly. Ovšem ze strany členské země potažmo státu je to problematický úkol, neboť stát jako takový neprovozuje žádný monitoring dopravní sítě nebo ve velmi omezené míře (dálniční mýtné brány-ty často provozuje soukromý subjekt, policejní kamerové systémy-není vždy vhodné pro zpracování dopravní dat, indukční smyčky), tzn. velice nízké pokrytí komunikací. Často se tedy jedná o stacionární zařízení, která měří dopravu a jejich nevýhodou je instalace pouze na vytížených komunikacích, nízká flexibilita a složitá instalace. Tudíž jsou tato data poptávána po soukromých subjektech (společnostech) a jsou považována společnostmi za velmi cenná. Tato data jsou velice užitečná pro státní správu, kdy jí data umožňují optimalizovat dopravu (tvořit inteligentní systémy, které umožňují řídit dopravu), nabízet aktuální dopravní informace nebo případně reagovat na dopravní kolapsy, nehody apod. Data jsou také užitečná pro statistické účely. Ale také s těmito daty má státní správa možnost tato data tzv. otevírat, čili je zveřejnit a vyhovět tak požadavku Evropské unie. Metody pro zisk těchto dat jsou následující [12] [13].

2.3.1 Pneumatický detektor

Jedná se o stacionární zařízení, které je umístěno na povrchu vozovky. Pracuje na principu změny tlaku v uzavřené hadici při průjezdu vozidla. Změna tlaku aktivuje spínač a ten vyšle signál do detektoru. Toto zařízení je limitováno počtem sledovaných koridorů. Nevýhodou je nízká spolehlivost v důsledku okolní teploty a snadné destrukce trubice. Výhodou je nízká pořizovací cena a rychlá instalace. Převážně se nasazuje na měřené úseky dočasně.

2.3.2 Piezoelektrický detektor

Jedná se o stacionární zařízení umístěné pod povrchem vozovky. Vyrábí se ve dvou provedeních trubkové či kabelové. Umisťuje se pro každý sledovaný pruh zvlášť. Výhodou je, že lze vozidla vážit za jízdy (tlakem kola se indukuje elektrické napětí) a snadno identifikovat jejich rychlost. Nevýhodou je, že pro instalaci nebo údržbu je nutno odklánět dopravu z jízdního pruhu. V případě instalace do nekvalitního povrchu cesty se mohou projevovat výpadky měření.

2.3.3 Indukční smyčky

Stacionární zařízení, které je uloženo pod povrchem vozovky. Jedná se o nejrozšířenější způsob měření dopravy. Jsou velmi jednoduché a spolehlivé. Indukční smyčky jsou tvořeny dráty, které jsou uloženy 30-60 mm pod povrchem vozovky ve tvaru čtverců nebo obdélníků. Při průtoku proudu vzniká magnetické pole, které je při přítomnosti automobilu narušeno. Na okraji silnice je zařízení, které detekuje narušení magnetického pole. Indukční smyčky umožňují získat data o intenzitě dopravy, rychlosti a rozestupy mezi vozidly. Nevýhodou je krátká životnost v důsledku deformace smyček, vyšší náklady na údržbu a odstranění poruch, nutnost odklonu dopravy při instalaci či údržbě. Výhodou je přesnost a spolehlivost.

2.3.4 Manuální sčítání

Jedná se o obvyklou metodu sběru dopravních dat. Princip je jednoduchý, umístí se pověřená osoba poblíž měřené komunikace. Klasickým vybavením jsou tužka a papír. Výhodou je možnost klasifikace vozidel do více skupin a možnost sledovat počet chodců.

2.3.5 Pasivní a aktivní infračervené detektory

Jedná se o stacionární zařízení, ovšem umístěné nad vozovkou. Aktivní infračervené detektory vysílají v infračerveném pásmu záření o nízkém výkonu, často pomocí LED diod. Paprsek je při odrazu od vozidla přijímán optickým senzorem. Pasivní infračervený detektor nevysílá žádné paprsky, ale zaznamenává změnu teploty vyzářenou průjezdem vozidla. Nevýhodou je, že kvalita (chyba) měření může být ovlivněna okolními podmínkami (déšť, mlha nebo sněžení) a mnohdy složitá kalibrace. Výhodou je možnost instalace bez zásahu do komunikace.

2.3.6 Mikrovlnný radar

Stacionární zařízení, které je umístěno nad vozovkou či mobilní zařízení (záleží na provedení). Princip měření je založen na šíření velmi krátkých elektromagnetických vln, kdy jsou vysílány velmi krátké impulzy o vysokém výkonu a odražené vlny jsou přijímány v pauzách. Vzdálenost vozidel se určuje dle časové korelace vyslaného a přijímaného signálu, kdy se využívá Dopplerova jevu. Normální umístění radaru by mělo svírat úhel přibližně 20 stupňů a neměl by být umístěn přímo v dráze vozidla. Jedna z velkých výhod krom měření rychlosti je kategorizace vozidel a měření intenzity vozidel. Další z výhod je použití na více jízdních pruzích a výsledky nejsou ovlivňovány nepřízní počasí. Nevýhodou je nemožná detekce stojícího vozidla, není doporučeno používat pro sčítání vozidel.

2.3.7 Ultrazvukové detektory

Stacionární zařízení, které je umístěno nad vozovkou. Princip měření je založen na vysílání tlakových zvukových vln, které jsou v rozsahu frekvence 25-50KHz, které jsou v pravidelných intervalech vysílány a měří se čas, který je potřebný k návratu vlny zpět do detektoru. Nevýhodou je nízká přesnost a možnost ovlivnění měření okolní teplotou nebo nepříznivými povětrnostními podmínkami. Výhodou je detekce počtu vozidel, výšky vozidla a možnost měření obsazenosti vozidel (přesnost může být negativně ovlivněna vysokou rychlostí vozidel).

2.3.8 Videodetekce

Stacionární zařízení, které je umístěno nad vozovkou či mobilní zařízení (záleží na provedení). Princip je založen na instalaci videokamery, která analyzuje obraz. Statický obraz je digitalizován a průjezd vozidla mění hodnoty barev a jasu, což umožňuje identifikovat a detekovat vozidlo. Výhodou je možnost používat kamery k více účelům. Nevýhodou je falešná detekce vozidel a nutnost údržby kamery (čištění čočky). Jedná se o jeden z velmi častých způsobů měření.

Častým způsobem měření je také kombinace výše uvedených detektorů, jenž může zvyšovat přesnost měření, či doplňovat měřené veličiny. Velmi častou kombinací je mikrovlnný radar, ultrazvuk a infračervené záření.

2.3.9 FCD (Floating Car Data)

Jedná se o mobilní zařízení, které je umístěno v automobilu. Toto zařízení je v posledních letech velice populární ve sběru dopravních dat a momentálně převažuje nad ostatními měřicími technikami, avšak FCD jsou používány jako doplňkový systém k běžným měřicím technikám. FCD je jedním ze základních stavebních kamenů Inteligentního dopravního systému (ITS), jehož cílem je zefektivnit dopravu, udělat dopravu bezpečnější a zvýšit její komfort. FCD jsou v porovnání s ostatními metodami měření dopravních dat méně nákladné na pořízení a údržbu [14].

Momentálně jsou dvě metody, které se nabízejí, GPS Floating Car Data (GFCD) a GSM Floating Car Data (CFCD). GFCD staví na dvou základních technologiích, globální satelitní

navigační systém (GNSS), který zajišťuje lokalizaci objektu a GSM (Global System for Mobile telecomunication), který zabezpečuje komunikaci s datovým centrem skrze SMS/GPRS [13]. Základní uvedené technologie jsou unifikované do FCD jednotky, která je umístěná ve vozidle. Původní záměr jednotky může sloužit ke správě vozového parku (fleet management), navigaci nebo monitorování odcizení vozidel. Pokrytí je nejlepší zejména ve městech, neboť systém je také využíván v prostředcích taxi služeb a prostředcích městské hromadné dopravy. Abychom získali věrohodná data na silnicích 1. třídy, rychlostní silnici a dálnici, je podmínkou průjezd 1 vozidla vybaveného FCD jednotkou minimálně do 15 minut na trase sledovaného úseku nebo ideálně 20 vozidel v 5 minutách (je vhodné, aby typy projíždějících vozidel byly rozlišné).

Druhou z metod je GSM Floating Car Data (CFCD), která staví na monitorování v buňkových sítích, konkrétně GSM (metoda je postavena pouze na GSM technologii) [13]. Principem je rozdělení dopravní sítě na úseky v závislosti pokrytí BTS(Base Transceiver Station), chceme-li základnovými stanicemi. V místě hranice základnové stanice je mobilní zařízení předáno jiné BTS, kde je vytvořen monitorovací bod. V buňkových sítích jsou různé metody, jak lokalizovat mobilní zařízení, uvedeme si pouze výčet těchto technik: Cell ID, Timing Advance, Enhanced Observed Time Difference, Angle of Arrival a Enhanced Cell Global Identity. Tato metoda nemusí být vždy přesná, co se týče lokalizace, avšak její nespornou výhodou je to, že pokud zvážíme možnost monitorování anonymních čísel, lze tedy sledovat mobilní telefony a v tom důsledku není třeba instalovat FCD jednotku do vozidla.

Abychom mohli využít dopravní data v aplikaci, je zapotřebí tato data zpracovat z měřících stanic. Standardní zpracování dat v případě FCD je následující, sběr dat (komunikace mezi FCD jednotkou a datovým centrem), tvorba dopravního obsahu (přiřazení dat z FCD jednotky, rozdělení do časových řad), integrace dopravních informací (výpočet kongesce a jízdního času), integrace do servisních služeb (informace pro řízení dopravy) a následné využití koncovým uživatelem. Co se týče dopravních dat, často je používán standard DATEX [16], což je standard pro inteligentní transportní systém pro Evropu, který uznal ústav pro Evropskou Technickou Specifikaci (European Technical Specification). DATEX je robustní řešení, které obsahuje nástroje, metodiky, definici procesů a pro nás důležitá XML schémata, jež slouží k výměně informací mezi dopravními informačními centry.

Zaměříme se pouze na Evropu. Jedna ze zemí, která poskytuje dopravní OpenData je Finsko. Za touto inciativou stojí Finská společnost InfoTripla. Bohužel v období, kdy vznikala tato práce, byla postupně prováděna migrace infrastruktury. Původně byl zdroj pro DATEX placený a byl (zatím ještě je v rámci kompatibility) dostupný druhý zdroj zdarma se SOAP protokolem. Nyní je spuštěna testovací verze s Web Feature Service standardem (spadá pod Open Geospatial Consortium) a není k němu zatím žádná dokumentace.

Zdroj se SOAP protokolem je velice strohý (placená verze s DATEXem obsahuje více informací), viz výpis 4, obsahuje ID měřící stanice, čas měření, počet vozidel a průměrnou rychlost. Není uvedeno, jakou metodou jsou data sbírána, předpokládáme, že indukčními smyčkami v kombinaci s FCD. Tento zdroj poskytuje CSV soubor, který obsahuje seznam měřících stanic, jejich

ID, název, kód komunikace (z kódu je možno dedukovat maximální rychlost v úseku) a souřadnice umístění (tento soubor byl použit jako zdroj měřících stanic pro generátor dopravních dat, kterému se budeme věnovat v poslední části). Tato OpenData používají licenci Creative Commons 4.0 BY (data lze sdílet, přizpůsobit a využívat komerčně, je nutno uvádět odkaz na licenci a uvádět, zda byly provedeny nějaké změny) [19]. Nevýhodou tohoto SOAP zdroje je nízké pokrytí komunikací (jsou pouze pokryty dálniční a rychlostní komunikace), ovšem v placené verzi je pokrytí lepší (síť komunikací je hustší). Pro základní analýzy co se týče rychlosti dopravy a plynulosti dopravy je postačující.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<LamDataResponse xmlns="http://tie.digitraffic.fi/sujuvuus/schemas">
<timestamp>
<utc>2016-04-20T19:10:00Z</utc>
<localtime>2016-04-20T22:10:00+03:00</localtime>
</timestamp>
<laststaticdataupdate>2014-09-24T09:59:27Z</laststaticdataupdate>
<lamdynamicdata>
<lamdata>
<lamid>555</lamid>
<measurementtime>
<utc>2016-04-20T19:04:00Z</utc>
<localtime>2016-04-20T22:04:00+03:00</localtime>
</measurementtime>
<trafficvolume1>4</trafficvolume1>
<trafficvolume2>2</trafficvolume2>
<averagespeed1>81</averagespeed1>
<averagespeed2>70</averagespeed2>
</lamdata>
<lamdata>
<lamid>556</lamid>
<measurementtime>
<utc>2016-04-20T19:04:00Z</utc>
<localtime>2016-04-20T22:04:00+03:00</localtime>
</measurementtime>
<trafficvolume1>9</trafficvolume1>
<trafficvolume2>8</trafficvolume2>
<averagespeed1>99</averagespeed1>
<averagespeed2>96</averagespeed2>
</lamdata>
```

Výpis 4: Ukázka XML výstupu z dopravního otevřeného zdroje InfoTripla [15]

Dalším ze zdrojů je Norsko, kde data spravuje Norská státní správa silnic. V Norské verzi stránek prezentují, že jejich data jsou OpenData (v anglické verzi není o tom ani zmínka) [17],

ovšem já tyto data za OpenData nepovažuji, neboť nejsou volně přístupná a o jejich přístup je nutné žádat skrze formulář [4]. Lze očekávat, že pokrytí nebude ideální, neboť data jsou sbírána pomocí mýtných bran (AutoPASS On Board Unit – elektronická dálniční známka) a videodetekce [18]. Opět je nutno konstatovat, že pro analýzu rychlosti dopravy a její plynulosti jsou data postačující. Vzhledem k tomu, že se podařilo získat přístup k DATEXu, můžeme si ukázat, jak vypadá XML výstup, viz výpis 5, pro získání cestovních časů. Element *Exchange* nám identifikuje, odkud data pochází. Element *PayloadPublication* obsahuje již data cestovních časů včetně dalších informací, kdy byla data publikována, zda se jedná o reálná data (*informationStatus*), či jestli data podléhají utajení (*confidentiality*). Element *elaboratedData* obsahuje samotná data, kde atribut *id* v elementu *predefinedLocationReference* obsahuje identifikátor úseku. Koordináty úseku jsou specifikovány v další části služby, včetně názvu úseku. Element *duration* pod elementem *travelTime* je cestovní čas ve vteřinách, jak dlouho trvá projet daný úsek. Element *freeFlowTravelTime* obsahuje informaci, v jakém průměrném čase bývá projížděn tento úsek, pokud v úseku není žádná doprava. V rámci Norského DATEXu jsou publikovány další informace krom dopravy, například informace o počasí (meteostanice), či seznam městských kamer.

```
<d2LogicalModel xmlns="http://datex2.eu/schema/2/2_0" modelBaseVersion="2">
<exchange>
<supplierIdentification>
<country>no</country>
<nationalIdentifier>Norwegian Public Roads Administration</nationalIdentifier>
</supplierIdentification>
</exchange>
<payloadPublication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="
    ElaboratedDataPublication" lang="nob">
<publicationTime>2016-04-05T13:20:37+02:00</publicationTime>
<publicationCreator>
<country>no</country>
<nationalIdentifier>Norwegian Public Roads Administration</nationalIdentifier>
</publicationCreator>
<headerInformation>
<confidentiality>noRestriction</confidentiality>
<informationStatus>real</informationStatus>
</headerInformation>
<elaboratedData>
<source>
<sourceCountry>no</sourceCountry>
<sourceIdentification>SVV</sourceIdentification>
</source>
<basicData xsi:type="TravelTimeData">
```

[4] webová adresa registračního formuláře http://www.vegvesen.no/en/Traffic/On+the+road/Datex2/get-access

```
<pertinentLocation xsi:type="LocationByReference">
<predefinedLocationReference targetClass="PredefinedLocation" version="1" id="100193"/>
</pertinentLocation>
<travelTimeTrendType>stable</travelTimeTrendType>
<travelTimeType>estimated</travelTimeType>
<travelTime>
<duration>579.0</duration>
</travelTime>
<freeFlowTravelTime>
<duration>573.0</duration>
</freeFlowTravelTime>
</basicData>
</elaboratedData>
<elaboratedData>
<source>
<sourceCountry>no</sourceCountry>
<sourceIdentification>SVV</sourceIdentification>
</source>
<basicData xsi:type="TravelTimeData">
...
```

Výpis 5: XML výstup Norského DATEXu

Podobný zdroj poskytuje i Švédsko. Data spravuje Švédská dopravní státní správa (Trafikverket). Data také nejsou volně dostupná a je nutno žádat o přístup skrze registrační formulář [5]. Opět je použit pro poskytování dat standard DATEX. Ke sběru dat je využito FCD a videodetekce [20]. Po delší reakci Trafikverkertu se také rovněž podařilo získat přístup k DATEXu. Ukážeme si fragment výstupu XML, viz výpis 6, se stejnou informací, jako u Norské dopravy. Zde můžeme vidět drobné odlišnosti od Norské verze, kde Švédsko publikuje více informací. Stejně jako u Norského DATEXu, i Švédsko publikuje další data. Nabízí rovněž počasí, ale i informace o náledí, o stavu vozovky či jednotlivých nehodách na komunikacích.

```
<d2LogicalModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
    www.w3.org/2001/XMLSchema" xmlns="http://datex2.eu/schema/2/2_0">
<exchange>
<supplierIdentification>
<country>se</country>
<nationalIdentifier>STA</nationalIdentifier>
</supplierIdentification>
</exchange>
<payloadPublication xsi:type="ElaboratedDataPublication" lang="sv">
<feedType>SRA_VSK</feedType>
```

[5] webová adresa registračního formuláře http://www.trafikverket.se/en/startpage/operations/Operations-road/Traffic-information/Real-time-traffic-information/How-to-subscribe—Real-time-traffic-information/registration/

```
<publicationTime>2016-04-05T12:51:23.2331155+02:00</publicationTime>
<publicationCreator>
<country>se</country>
<nationalIdentifier>VSK</nationalIdentifier>
</publicationCreator>
<headerInformation>
<confidentiality>noRestriction</confidentiality>
<informationStatus>real</informationStatus>
</headerInformation>
<elaboratedData>
<source>
<sourceCountry>se</sourceCountry>
<sourceIdentification>VSK</sourceIdentification>
</source>
<basicData xsi:type="TravelTimeData">
<measurementOrCalculationPeriod>300</measurementOrCalculationPeriod>
<measurementOrCalculationTime>2016-04-05T12:44:07+02:00</measurementOrCalculationTime>
<pertinentLocation xsi:type="LocationByReference">
<predefinedLocationReference id="5001" version="0" targetClass="PredefinedLocation"/>
</pertinentLocation>
<travelTimeTrendType>stable</travelTimeTrendType>
<travelTimeType>instantaneous</travelTimeType>
<travelTime>
<duration>40</duration>
</travelTime>
<freeFlowTravelTime>
<duration>39</duration>
</freeFlowTravelTime>
<normallyExpectedTravelTime>
<duration>0</duration>
</normallyExpectedTravelTime>
<freeFlowSpeed>
<speed>0</speed>
</freeFlowSpeed>
</basicData>
</elaboratedData>
<elaboratedData>
<source>
<sourceCountry>se</sourceCountry>
<sourceIdentification>VSK</sourceIdentification>
</source>
```

Výpis 6: XML výstup Švédského DATEXu

Se ziskem dat je situace úplně stejná jak u Anglie, tak i u Francie a Španělska [16]. Ovšem přístup se nepodařilo získat.

Jedním ze zdrojů je Německé město Kolín nad Rýnem, je potěšující, že město opravdu poskytuje OpenData, která se týkají dopravních informací v reálném čase, ale bohužel je poskytnut pouze index využití komunikace [6]. Tudíž pro analýzu dopravy není úplně vhodný. Licencování je Creative Commons, ale jedná se o německou adaptaci licence (CC BY 3.0 DE), v podstatě jde o starší verzi Open licence, kterou jsme si uvedli u Finských OpenData [19].

Zajímavý projekt, který stojí za zmínku je projekt OpenTraffic [7]. Bohužel se zatím jedná o pilotní projekt, ale měl by právě zpřístupnit nedostupná otevřená data. Tento projekt kooperuje s projektem Open Street Map, jenž poskytuje geografická OpenData (mapové podklady). Mělo by se jednat o zisk dat z FCD.

Situace kolem dopravních OpenDat není příznivá. Je to hlavně z důvodu problematického zisku dat a často vysoké ceny dat. Dá se očekávat, že tato situace se v následujících letech bude výrazně zlepšovat. Dobrou zprávou je, že je snaha poskytovaná dopravní data standardizovat standardem DATEX a proto nemusí být potřeba transformovat data z různých formátů do jednotného formátu. Avšak další nevýhodou je nemožnost získat dopravní data ze všech států Evropské unie.

[6] webová adresa pro OpenData města Kolína nad Rýnem http://www.stadt-koeln.de/externe-dienste/open-data/traffic.php

[7] webová adresa projektu OpenTraffic http://opentraffic.io

3 Analýza možností poskytování dat ze zadaných datových zdrojů

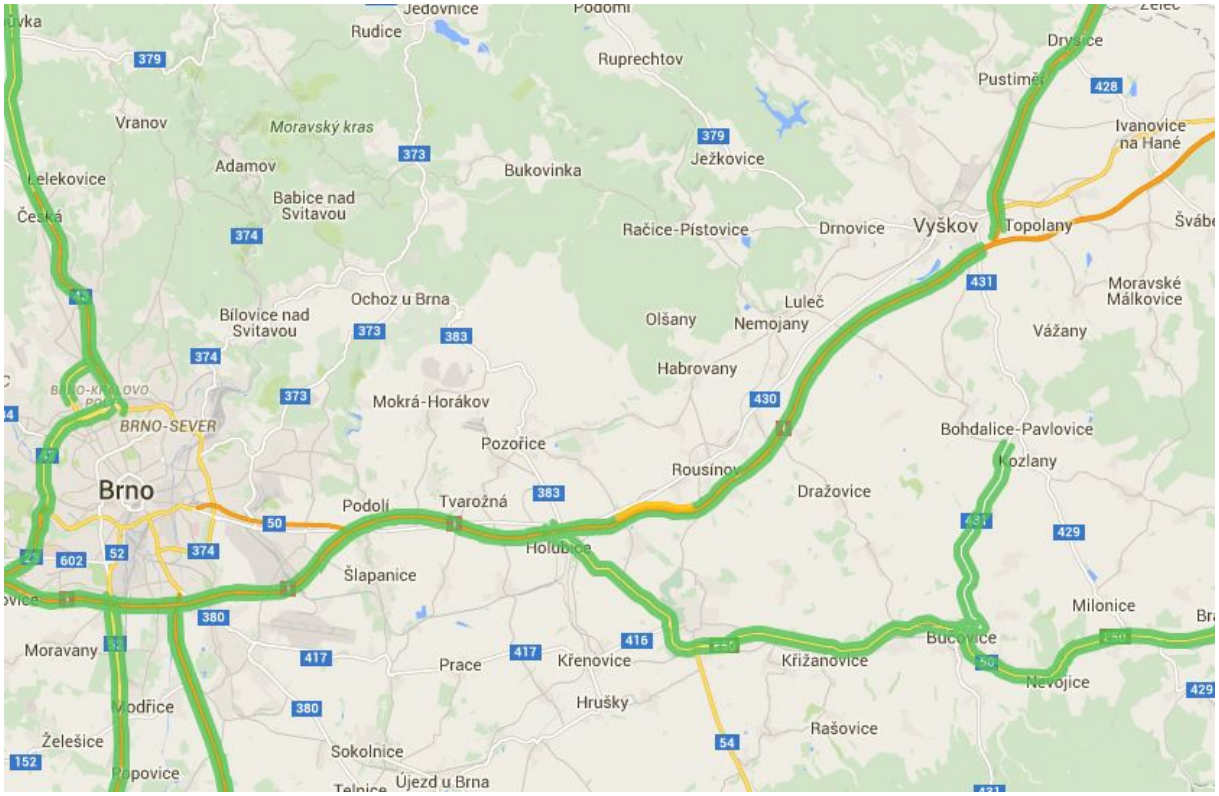
Další možností je prozkoumání komerční sféry poskytování dopravních informací. Nabízí se 3 největší firmy v poskytování dopravních dat a těmi jsou INRIX, Here a TomTom. Tyto firmy vykupují data, jak ze státního sektoru, tak ze sektoru soukromého (taxi služby, fleet služby apod.). V případě všech jmenovaných firem využívají své produkty ke sběru dat. Díky těmto aktivitám mohou nabídnout data i ve státech, kde nejsou dostupná dopravní OpenData, ale obecně i dopravní data. Společnosti na poskytování dopravních dat jako jsou Garmin a Waze nebudeme rozebírat, neboť částečně (z poloviny) využívají data, která poskytuje INRIX [21].

U komerčních řešení je velmi populární protokol TPEG (Transport Protocol Experts Group), což je sada datových protokolů pro dopravní informace. Tento protokol je často využíván v buňkových sítích, neboť jeho výhodou oproti DATEXu je nízká náročnost z pohledu objemu dat [22]. Ač by se dalo chápat, že DATEX a TPEG jsou konkurenční řešení, není tomu tak. Obě tato řešení můžou být mezi sebou propojena a úzce spolupracovat. Tento protokol často využívá pro přenos dat standardů Digital Audio Broadcasting (DAB) a Digital Multimedia Broadcasting (DMB). Další technologie, která se ještě stále používá je RDS-TMC (Radio Data System - Traffic Message Channel) [8], což je systém k přenosu informací skrze velmi krátké vlny rádiového FM vysílání. Implementace TPEG protokolu částečně vychází z RDS-TMC. U protokolu TPEG vznikají nejasnosti v označení, neboť za lomítkem se může uvádět, k jakému účelu (aplikaci) protokol slouží (WEA – předpověď počasí, TFP – dopravní data, TEC – dopravní události) nebo jakou technologií je protokol přenášen (GSM, DAB).

3.1 INRIX

Jedná se o firmu, která byla založena v USA v roce 2004. INRIX poskytuje řadu internetových služeb vztahující se k dopravě [23]. V nabídce produktů jsou 3 produkty. INRIX Insights, což je produkt, který se zaměřuje na analýzu dat, usnadňuje plánování nových cest a tranzitních systémů, ale i hledání vhodného místa k podnikání nebo marketingové kampani. Dalším produktem je INRIX Connected Car Services, který pomáhá výrobcům automobilů zvýšit komfort navigování v palubních systémech (služba umí vyhledávat volná parkoviště, hledání vhodné trasy, vyhnout se zácpám, poskytovat informace o počasí, poskytování zájmových bodů). Posledním produktem je INRIX XD Traffic, což je služba, která poskytuje prediktivní dopravní data a dopravní data v reálném čase. Klíčovým produktem je produkt INRIX Traffic App, což je mobilní aplikace, která poskytuje mapové podklady a dopravní informace (aplikace je srovnatelná s aplikací Waze), taktéž posílá dopravní informace INRIXu. INRIX využívá různých způsobů zisku dat, například FCD (CFCD i GFCD), zisk dat z třetích stran apod. INRIX úzce spolupracuje s výrobci automobilů a propojuje svůj software za účelem poskytování navigace, ale i k zisku dat (Ford SYNC, Toyota Entune, Audi, BMW ConnectedDrive). INRIX využívá protokol

[8] http://www.rds-tmc.cz



Obrázek 5: Náhled na INRIX interaktivní mapu

TPEG/GSM k zisku dat a pro komunikaci s INRIX API je použit protokol SOAP. Pro nás je důležité, že INRIX poskytuje dopravní data v reálném čase, ale nabízí i data historická. INRIX jako jediná komerční služba v rámci svého API nabízí API pro informace o parkování, API pro informace o čerpacích stanicích a API pro databázi dopravních kamer. Na základě emailové komunikace je cena za stát na měsíc 7500€, jedná se o cenu pro výzkum (cena k 9. 11. 2015). INRIX nabízí velice slušné pokrytí, jedná se o USA, většinu států EU a vybrané města v Asii. Vzhledem k horší emailové komunikaci s INRIXem se nepodařilo získat přesný popis API, přesný popis dat ani licenční podmínky k poskytování dopravních dat. Podařilo se pouze získat přístup k interaktivním mapovým podkladům, viz obrázek 5.

3.2 TomTom

Firma založena roku 1991 v Nizozemsku. Firma se zabývá navigačním softwarem a navigačními přístroji, rovněž poskytuje služby ohledně poskytování dopravních dat (TomTom Traffic, TomTom Live, TomTom HD Traffic) [24]. Tak jako INRIX i TomTom se angažuje do automobilového průmyslu (Renault R-Link, Mazda Navigation Systém, Toyota Aygo Connect Multimedia). Díky svým aktivitám v automobilovém průmyslu a výrobě navigačního softwaru a hardwaru má výbornou základnu pro sběr dat. Pomocí svých online placených služeb, které přidávají hodnotu k TomTom produktům sbírá data, příkladem je TomTom PLUS, TomTom LIVE Services. Sa-

mozřejmě i TomTom vykupuje data od soukromých či státních organizací, ale rozhodně v menší míře, než konkurence. TomTom jako jeden z mála komerčních služeb poskytuje zkušební omezený přístup, jedná se o 10 000 požadavků denně a 900 požadavků za vteřinu. Cenu placené služby se bohužel nepodařilo zjistit. API pro poskytování dopravních dat v reálném čase používá REST a pro zisk informací používá protokol TPEG/GSM, DATEX a Protobuf. TomTom poskytuje dopravní historická data. V rámci licenčních podmínek je zakázáno provádět jakékoliv dodatky, modifikace, úpravy nebo jiné změny licencovaných produktů, přidávat data nebo je kombinovat s licencovanými produkty, nebo provádět reverzní inženýrství, dekompilace nebo dělat derivát prací. Držitel licence není oprávněn sestavit databázi pomocí extrakcí nebo odčerpáváním licencovaných produktů v kombinaci s jakoukoli jinou databází licence nebo jakékoliv třetí strany, což vyplývá z předchozí věty a to, že data nesmí být kombinována. Zajímavostí je, že TomTom má ve smlouvě seznam firem, kterým nesmí být API poskytnuto. Mezi firmami je například Amazon, Google, Easy Taxi apod. Další zajímavostí je, že data, která TomTom získal v Číně a v Korei nesmí být poskytnuta mimo hranice.

3.3 Here

Za vznikem firmy Here stojí společnost Nokia, která koupila společnost Navteq, která se zabývala mapovými podklady a geografickými systémy [25]. V roce 2015 Nokia prodala společnost Here skupině automobilových firem (VW koncern, BMW, Daimler). Tudíž momentálně firma sídlí v Německu. Firma Here se zaměřuje na tvorbu mapových podkladů a poskytování služeb ohledně mapových podkladů. Here spolupracuje se společnostmi Alpine, Garmin, BMW, Oracle, Amazon a Microsoft. Společnost poskytuje stejnojmennou aplikaci pro mobilní platformu jako navigační software. Here poskytuje své mapové podklady společnostem, jako jsou Facebook, Bing a nebo Yahoo! Maps. Díky široké základně partnerů a své navigační aplikaci není získávání dopravních dat žádný problém. Společnost Here má 3 hlavní oddělení. Oddělení Spotřebitel (Consumer) se věnuje vývoji a údržbě Here aplikace. Oddělení Společnost (Enterprise) se věnuje poskytování API, služeb a mapového obsahu. Oddělení pro automobilový průmysl (Automotive) se zaměřuje na integraci navigačních systémů do automobilů a tvorbě inteligentních aut. Ze všech firem, které jsem kontaktoval, byla firma Here nejsdílnější ohledně informací. Pro zisk informací používá Here pouze protokol TPEG/GSM. API je přístupné pomocí REST API a JavaScriptu. Here nabízí několik tarifů k přístupu k API. Rozlišuje se, zda se s daty bude nakládat komerčně či nikoliv. Rozdíl mezi komerčním a spotřebitelským přístupem se liší pouze v ceně a počtem transakcí (přístupů do API). V nabídce jsou 3 plány, Starter, Standard, Advanced [31]. V následující tabulce 1 je uvedena cena a přehled plánů, jako první je uvedena hodnota pro spotřebitele, jako druhá hodnota je určena pro komerční účely.

Here v rámci svého API poskytuje zkušební přístup na 3 měsíce, kde je dostupných 100 000 transakcí, což je velmi velkorysé ze strany firmy Here a poskytuje veškerý přístup ke všem API. V následujícím textu bude osvětleno, k čemu různá API slouží. Map Tile API slouží k získání mapových satelitních a leteckých snímků ve formě obrázku. Map Image API nám umožňuje

Tabulka 1: Přehled tarifů společnosti Here

	Starter	**Standard**	**Advanced**
Cena za měsíc (9. 11. 2015)	49€/149€	99€/399€	299€/1 199€
Počet transakcí	150 000/50 000	275 000/100 000	700 000/250 000
Map Tile API	A	A	A
Map Image API	A	A	A
Geocoder API	A	A	A
Places API	A	A	A
Weather API	A	A	A
Routing API	N	A	A
Venue Images	N	N	A
Incidents	N	N	A
Traffic API	N	N	A

získat generovaný mapový podklad společnosti Here. Geocoder API slouží k převodu z textové adresy na geografické souřadnice a naopak. Places API slouží k zisku zájmových bodů včetně uživatelských recenzí, fotek, popisů, adresy a kategorie. Weather API slouží k zisku informací o počasí. Routing API slouží k zisku informací ohledně plánování cesty, itineráře, preference naplánované cesty (krátká nebo rychlá), zákazy, mýtné, čas příjezdu a souřadnice cesty. Venue Images nám dává přístup k databázi obrázků míst v různých výškových úrovních. Incidents je API pro sběr událostí na dané komunikaci (nehoda, uzavřená komunikace apod). Traffic API, pro nás nejdůležitější API, slouží k zisku dopravních informací, jako je například průměrná rychlost na komunikaci, úroveň zácpy atd. Velmi pozitivní zprávou je, že Here má speciální plán i pro univerzity, který může využívat na škole jakýkoliv student a učitel. Počet transakcí je 1 200 000 a cena za rok je 42 495€. Cena je sice vyšší, ale škola si s daty může nakládat, podle libosti (jen je nesmí přeprodávat). Pokud by se cena srovnala s INRIXem, tak je stále nízká, neboť s tarifem, ať už veřejným nebo univerzitním, získáváme přístup ke všem státům, kde má Here pokrytí, kdežto cena u INRIXu byla uvedena za stát. Nejdůležitější licenční podmínky z pohledu programátora jsou následující, aplikace by neměla být navržena pro systémy s vysokým rizikem (aplikace související s lidským zdravím nebo ohrožující bezpečnost lidí) [30]. Je zakázáno pozměňovat či nějak upravovat poskytovaná data. Také je zakázáno vytvářet kombinaci služeb či vylepšené služby, které by mohly vést k tvorbě derivátu lokalizační platformy nebo obsahu, který by mohl být podobný a konkurenční k produktu či službám Here. Také je zakázáno tvořit službu, která by mohla degradovat kvalitu Here materiálů. Jakékoliv reverzní techniky Here služeb a aplikací jsou zakázány. Je rovněž zakázáno vykonávat jakékoliv zátěžové testy, které by mohly ovlivnit běh Here služeb.

Můžeme vidět přehledovou tabulku 2, která porovnává nabídku komerčních poskytovatelů dopravních dat. Ač každý poskytovatel má pro svou službu jiný název, nabízená služba poskytuje stejné možnosti nebo podobné jako u konkurence. Názvy služeb byly použity převážně od poskytovatele Here. Co služby poskytují, jsme si krátce popsali o pár odstavců výše. Názvy služeb,

Tabulka 2: Srovnávací tabulka jednotlivých API komerčních služeb

	INRIX	**TomTom**	**Here**
Map Tile API	A	A	A
Map Image API	A	N	A
Venue Images	N	N	A
Routing API	A	A	A
Geocoder API	N	A	A
Places API	A	N	A
Traffic API	A	A	A
Weather API	A	N	A
Incidents	A	A	A
Parking Info	A	N	N
Traffic Cam	A	N	N
Gas Stations Info	A	N	N

které jsme si nepopsali, jsou Parking Info. To jest API, které poskytuje databázi parkovišť a informace o volných místech na parkovištích. Traffic Cam je API poskytující databázi, informace a adresy kamer, jež monitorují danou komunikaci. Gas Station Info je API poskytující přehled benzínek, jejich umístění a ceny.

Uvedeme si pro zajímavost ještě tabulky, viz příloha B, jež obsahují seznam zemí v Evropské unii, které pokrývá nějaká komerční služba. Země, které nejsou zmíněny v tabulce, nemají žádné komerční pokrytí. Příkladem je Bosna a Hercegovina. Z tabulky můžeme vyčíst, že v pokrytí většiny členských státu EU, kraluje trojice INRIX, TomTom a Here. Můžeme si všimnout, že v tabulce se občas vyskytuje společnost Be-Mobile, což je Belgická společnost, ovšem do analýzy jsme ji nezahrnuli, neboť má velice nízké pokrytí, ovšem pokrývá státy, které nepokrývá jiná komerční služba.

Vzhledem k tomu, že Here poskytuje 3 měsíční zkušební dobu k přístupu do svého API a zisk této zkušební verze nebyl nijak zvláště komplikovaný, na rozdíl od INRIXu, po zbytek této práce budeme většinou pracovat s datovou strukturou, kterou používá Here Traffic API. Společnost Here jsme si vybrali i z toho důvodu, že mají volně přístupnou dokumentaci (tedy až na popis datové struktury, ta je dostupná po přihlášení) a vysoké pokrytí států EU.

3.4 Analýza vybrané datové struktury

Máme vybraný zdroj, dále si stáhneme měsíční data např. pro Prahu a zanalyzujeme datovou strukturu. Získali jsme API ID a API kód a máme volných 100 000 požadavků na měsíc. Pro stažení dat si napíšeme jednoduchý program (HereDownloader) v .NETu, konkrétně v C#, který přistupuje do Here API pomocí parametrizované URL adresy. Rozhodneme, jakým způsobem budeme vybírat body z mapy. Zda si vybereme Mercator projekci, která mapuje body povrchu zeměkoule na body v rovině nebo použijeme tzv. Quadkey (Quadtree klíče), což je stromová datová struktura, která je podobná binárnímu stromu, ale místo dvou ukazatelů má ukazatele

čtyři (jeden pro každý kvadrant). Každý Quadkey je unikátní pro jednu dlaždici v mapě. Ač je Quadkey velice zajímavý systém pro určení polohy na mapě, zvolíme tradiční Mercator projekci [34] z důvodu jednoduché implementace. Abychom zjednodušili stahování dat, program si na svém začátku vyžádá adresu, která bude sloužit jako centrální bod. Pomocí Google Geocoding API (které je veřejné), ze zadané adresy získáme GPS souřadnice. Dále je nutné si vyžádat míru přiblížení mapy, tedy oblast, která má být zachycena. Poté máme všechny informace k převedení GPS adresy do Mercatorova zobrazení. Po převodu zbývá již jen zvolit velikost kroku, mezi jednotlivými intervaly, kdy se mají stáhnout data. Pro usnadnění práce s Here API, které používá REST, použijeme knihovnu RestSharp, která velmi usnadňuje práci a manipulaci s REST API. Při volání API si ještě do parametru uvedeme, že chceme vrátit výsledek v XML formátu (je možnost získat výsledek i ve formátu JSON). Získaná XML data následně ukládáme, viz ukázka ve výpise 7. Na výsledku uloženého souboru si popíšeme datovou strukturu, kterou používá Here.

```
<TRAFFICML_REALTIME xmlns="http://traffic.nokia.com/trafficml-flow-3.1" MAP_VERSION=
    "" CREATED_TIMESTAMP="2015-10-25T21:43:20Z" VERSION="3.1" UNITS="metric">
  <RWS TY="TMC" MAP_VERSION="201503" EBU_COUNTRY_CODE="2"
      EXTENDED_COUNTRY_CODE="E2" TABLE_ID="25" UNITS="metric">
    <RW LI="225+00302" DE="Jizni spojka" PBT="2015-10-25T21:42:29Z" mid="c45ffa33-8b84
        -4f55-929b-7c35c9ab9d1c|">
      <FIS>
        <FI>
          <TMC PC="4110" DE="Modranska-Branik" QD="-" LE="0.8101" />
          <CF CN="0.82" FF="72.4" JF="0.0" SP="76.17" SU="76.17" TY="TR" />
        </FI>
        <FI>
          <TMC PC="1377" DE="V podzamci" QD="-" LE="1.99232" />
          <CF CN="0.97" FF="72.4" JF="0.0" SP="80.0" SU="82.1" TY="TR" />
        </FI>
        <FI>
          <TMC PC="1379" DE="Videnska" QD="-" LE="1.2877" />
          <CF CN="0.98" FF="72.4" JF="0.0" SP="80.0" SU="87.24" TY="TR" />
        </FI>
        <FI>
          <TMC PC="1381" DE="5. kvetna" QD="-" LE="0.84023" />
          <CF CN="0.83" FF="72.0" JF="0.0" SP="76.23" SU="76.23" TY="TR" />
        </FI>
        <FI>
          <TMC PC="1383" DE="Sporilovska" QD="-" LE="1.58341" />
          <CF CN="0.9" FF="72.4" JF="0.68399" SP="68.5" SU="68.5" TY="TR" />
        </FI>
        <FI>
          <TMC PC="1385" DE="V korytech" QD="-" LE="1.71739" />
          <CF CN="0.99" FF="72.4" JF="0.0" SP="78.25" SU="78.25" TY="TR" />
```

```
    </FI>
    <FI>
      <TMC PC="1387" DE="Svehlova" QD="-" LE="0.60679" />
      <CF CN="0.97" FF="72.4" JF="0.0" SP="79.63" SU="79.63" TY="TR" />
    </FI>
    <FI>
      <TMC PC="23085" DE="Lanovy most" QD="-" LE="0.76115" />
      <CF CN="0.98" FF="71.0" JF="0.0" SP="78.8" SU="78.8" TY="TR" />
    </FI>
  </FIS>
</RW>
<RW LI="225-00302" DE="Jizni spojka" PBT="2015-10-25T21:42:41Z" mid="37aff944-0a3b
    -4ab3-8f71-1c7a31d48980|">
```

Výpis 7: XML výstup Here Traffic API

Jeden dotaz do API nám vrací kořenový element *TRAFFICML_REALTIME*, který má atributy *MAP_VERSION*, který nemusí být vždy vyplněn, ale značí verzi mapových podkladů, dále atribut *CREATED_TIMESTAMP*, který slouží k poskytnutí data, kdy bylo o data zažádáno [26] [27]. Dále je atribut *VERSION*, který značí verzi Traffic API a poslední atribut *UNITS*, který označuje, v jakých jednotkách jsou data vyjádřena (metrické nebo imperiální) [28]. Element *RWS* (roadway items) nám uvádí seznam cest, ten má atributy *TY*, který značí poskytnutí *TMC* (Traffic Message Channel) elementu (pokud nabývá jiné hodnoty, než *TMC*, mají se data ignorovat), dále atribut *MAP_VERSION*, který jsme si osvětlili. Atribut *EBU_COUNTRY_CODE* obsahuje kód země z *TMC* tabulky (uvádí se z důvodů zpětné kompatibility). Atribut *EXTENDED_COUNTRY_CODE* obsahuje kód země dle tabulky služby Here. Atribut *TABLE_ID* uvádí kód tabulky, kde se mají hledat *EBU_COUNTRY_CODE* a atribut *UNITS* jsme si uvedli. Element *RW* (roadway) nám uvádí jeden konkrétní úsek, který je měřen. Ten má atributy *LI* (linear identifier), což je unikátní identifikátor pro úsek, číslo před znaménkem je složeno z *EBU_COUNTRY_CODE* a *TABLE_ID*, pak následuje znaménko, které značí směr úseku (kladný nebo záporný) a identifikátor úseku. Atribut *DE* (description) obsahuje textový popis úseku, dále atribut *PBT* obsahuje datum měření a atribut mid obsahuje Here identifikátor. Element *FIS* (flow item elements) obsahuje seznam částí úseku (ve městech je úsek většinou rozdělen po křižovatkách). Dále element *FI* (flow item) obsahuje podelementy, které popisují informace dané části úseku. Element *TMC* obsahuje metadata části úseku, která jsou vyjádřena v atributech. Atribut *PC* je *TMC* lokační kód (opět je uveden z důvodu zpětné kompatibility), atribut *DE* jsme si popsali (obsahuje popisek úseku). Atribut *QD* (queuing direction) obsahuje směr úseku (opět kladný nebo záporný) a atribut *LE* (length) obsahuje délku úseku od předchozího bodu (v případě první části úseku je samozřejmě hodnota předchozího bodu nula), pokud máme metrické jednotky, hodnota je uváděna v kilometrech. Ještě se může vyskytnout jeden neobvyklý atribut a tím je atribut *SN*, ten nese informaci o tom, kdy byla provedena změna dat, jde o jakýsi ekvivalent atributu *PBT*. Může se ovšem stát, že informace na každé části úseku

byly získány v různé doby, pak tento atribut má smysl. Přichází na řadu nejdůležitější element a tím je element *CF* (current flow). Obsahuje atribut *CN* (confidence), který značí důvěryhodnost dat vyjádřenou v procentech (pokud hodnota je -1, tak část úseku je uzavřena). Atribut *JF* (jam factor) značí úroveň zácpy na části úseku, hodnota se pohybuje v intervalu od 0 do 10, hodnota -1 značí, že úroveň zácpy nemůže být spočítána. Další atribut je *FF*, ten nám dává informaci, jaká je rychlost průjezdu, pokud cesta je úplně volná (free flow). Tato hodnota je měřena, pokud úroveň zácpy je nula, jedná se o průměrnou hodnotu. Atribut *SU* (speed unit) je aktuální rychlost na části úseku, dále atribut *SP* je aktuální rychlost na části úseku, ale omezená rychlostním limitem (je třeba poznamenat, pokud se jedná o úsek, kde je proměnlivá rychlost, pak tato data nemusí být správná). Při bližším zkoumání si můžeme všimnout, že hodnota *FF* je nižší, než hodnota *SP* nebo *SU*. Podařilo se mi kontaktovat vývojový tým, který se zabývá Traffic API a bylo mi podáno vysvětlení, že *FF* může být nižší z toho důvodu, aby hodnota seděla do rovnice, pomocí které se počítá úroveň zácpy. Zároveň je třeba vzít v potaz také fakt, který již byl zmíněn, že předepsaná rychlost na úseku se může měnit. Posledním z atributů je atribut *TY*, který je získán pomocí lokační reference, v drtivé většině případů obsahuje hodnotu *TR*. Pak se vyskytují v XML výstupu nepříliš časté elementy a atributy. Jedním z nich je element *PF* (predictive flow), tedy jedná se o element, který obsahuje předpovědní informace. Obsahuje dva atributy *UT* a *SP*. *UT* definuje čas, po který je predikce platná a *SP* obsahuje predikovanou rychlost, skrze tuto hodnotu lze sledovat předpokládaný vývoj rychlosti na daném úseku. Posledním elementem je element *SHP* (shapes), který má jediný atribut *TC*. Ten obsahuje počet souřadnic, které definují křivky úseku [29].

Vzhledem k tomu, že v další části práce budeme řešit ukládání a práci s těmito daty, výsledná datová struktura, se kterou budeme pracovat, bude obsahovat tyto atributy: *CREATED_TIMESTAMP*, *EBU_COUNTRY_CODE*, *TABLE_ID*, *LI*, *DE*, *PBT*, *PC*, *LE*, *SN*, *CF_CN*, *CF_FF*, *CF_JF*, *CF_SP*, *CF_SU*, *CF_TY*, *PF_UT*, *PF_SP* a *SHP_FC*. Atributy, které chybí v definované struktuře, mohou být zpětně získány a není třeba je ukládat nebo nejsou důležité. Atributy, které obsahují časové informace, jsou převáděny z formátu DateTime (ISO 8601) [33] na formát Unix Timestamp. Tento formát byl zvolen z důvodu úspory místa, neboť Unix Timestamp potřebuje 4 byty k uložení, kde postačuje datový typ integer. Kdežto DateTime je komplexní datový typ a spotřebuje více bytů (obvykle 8 bytů, záleží na použité platformě). Unix Timestamp má i své nevýhody, jednou z nich je, že datum a čas není běžně čitelný z hodnoty a v roce 2038 dojde k přetečení hodnoty, ovšem pouze na 32bitových systémech.

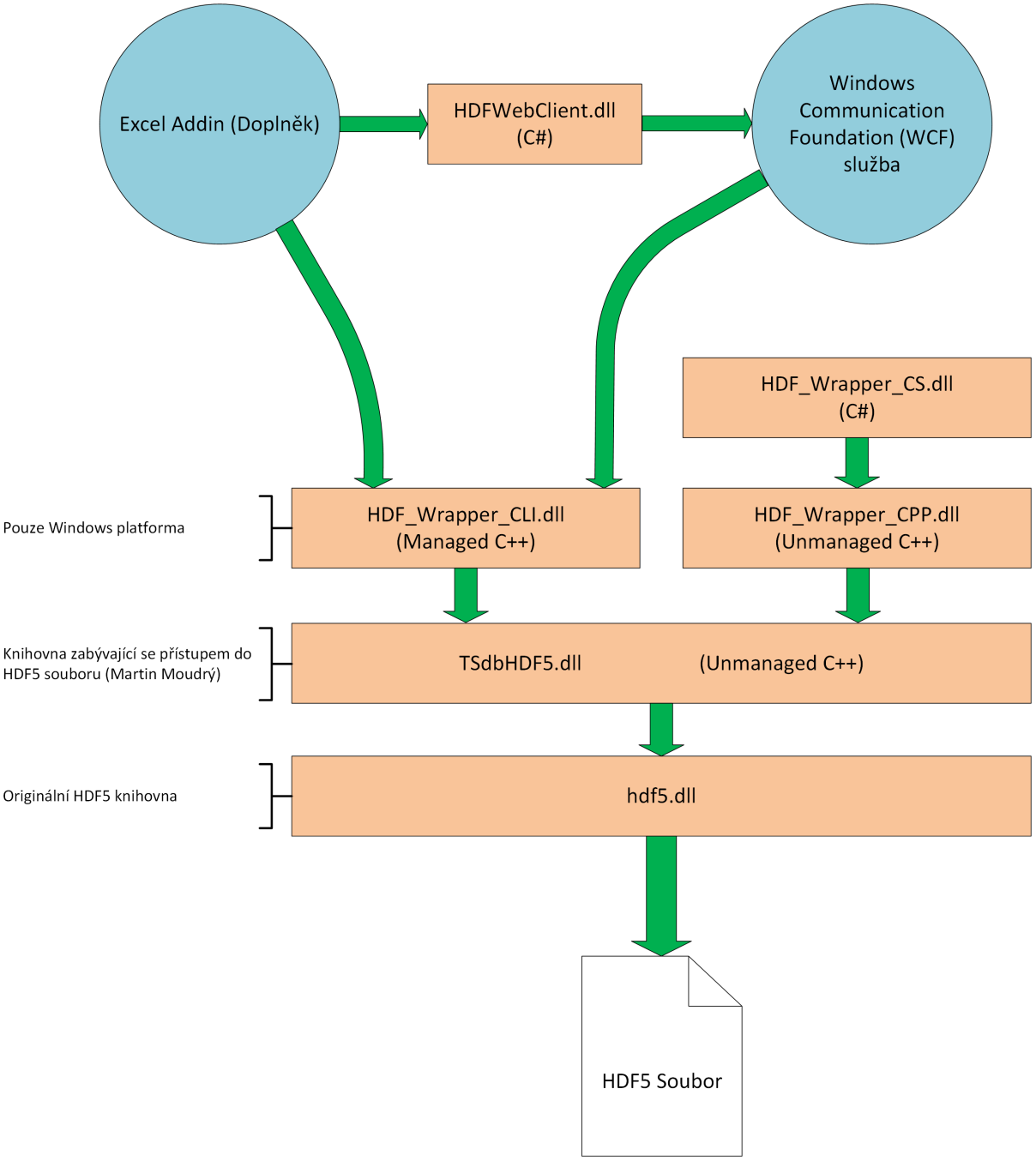
4 Implementace adaptéru pro čtení dat pro výpočty na HPC clusteru

Implementace adaptéru je postavena na diplomové práci Martina Moudrého [36], jenž se zabývá rozšířením HDF5 o ukládání časových řad (práce v neposlední řadě zabývá formátem HDF5). HDF5 je hierarchický datový formát určený pro ukládání komplexních a objemných dat (Big Data) [37]. Jeho výhodou je využívání univerzálního datového modelu, který může obsahovat komplexní datové objekty a širokou škálu metadat. Další jedna z výhod je dostupnost nástrojů a aplikací pro manipulaci s formátem HDF5. Nespornou výhodou, co se týče IO (vstupních/výstupních) operací s diskem, je vysoký výkon formátu HDF5 oproti jiným metodám, jak ukládat data. Výhodou je taktéž podpora komprese [38]. Vzhledem k povaze diplomové práce, je cílem ukládat časové řady orientované nejen na dopravu. Zadáním je tedy vytvořit jakousi abstrakci nad formátem HDF5, která umožní velice snadnou implementaci nových datových struktur. Cílem této kapitoly je popsat vzniklou architekturu.

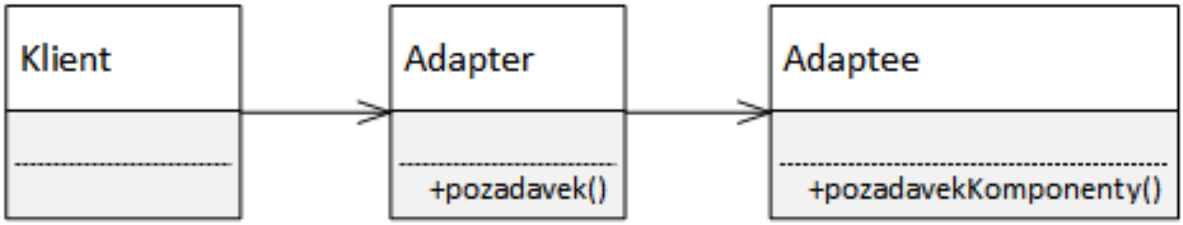
Jedním z dalších požadavků bylo vytvořit webovou službu, klientskou knihovnu využívající webovou službu, doplněk pro Excel (bude mu věnována samostatná kapitola) a samozřejmě adaptér (wrapper). Architektura byla vyvíjena tak, aby ji bylo možno provozovat pod operačním systémem Windows a Linux. Tato architektura má nespornou výhodu v tom, že pokud se změní některá z nižších vrstev, tak vyšší vrstvy budou stále funkční bez potřeby měnit stávající kód (neplatí pouze v případě manipulace s datovými strukturami). Jde hlavně o to, pokud se někdo rozhodne, že nechce používat formát HDF5 nebo se API formátu HDF5 radikálně změní, upraví se pouze adaptéry. Celá architektura je postavena pouze na 64 bitové platformě. Pro systém Linux je potřeba běhového prostředí Mono pro .NET prostředí a ostatní knihovny zkompilované pomocí kompilátoru GCC nebo Intel (zde uvedené kompilátory jsou ověřené), pro ostatní kompilátory není garantována funkčnost.

4.1 Adaptér pro řízené `C++`

Uvedeme si knihovnu (HDF_Wrapper_CLI.dll) pro řízené (spravované, Managed) `C++`, která přistupuje do knihovny TSdbHDF5.dll (v této kapitole označena jako nižší vrstva). Tato knihovna vznikla jako první z architektury a využívá programovací jazyk `C++/CLI` (Common Language Infrastructure), který je standardizován společností ECMA. Cílem jazyka je spojit svět .NETu a `C++`, tedy jazyk `C++`, který využívá CLR (Common Language Runtime). Bohužel jeho jedinou nevýhodou je, že není portován platformou Mono, která by umožnila funkcionalitu na systému Linux a s podporou open source oficiální implementací .NET Core se také nepočítá. Tato knihovna je využívána Excel doplňkem a HDF5 webovou službou (WCF). Obecně doporučuji preferovat tuto knihovnu na operačním systému Windows, už vzhledem k tomu, že se volá pouze jedna knihovna, namísto knihoven dvou. Z pohledu testování softwaru a platformy .NET se lépe testuje `C++/CLI`, než `C++`, neboť pro `C++/CLI` existuje přímá podpora testů. Pro `C++` se musí vy-



Obrázek 6: Architektura vzniklých knihoven



Obrázek 7: Návrhový vzor Adapter [35]

tvářet samostatný testovací projekt, který se musí přepnout do módu CLR, tedy pokud chceme testovat knihovnu HDF_Wrapper_CPP.dll napřímo.

Adaptér obaluje veškerou funkcionalitu knihovny nižší vrstvy. Adaptér navenek pracuje s objekty, ale nižší vrstva pracuje se strukturami, tudíž je jedním z úkolů konvertovat struktury na objekty. Je to hlavně z důvodu kompatibility s jazyky, které by využívaly adaptér a hlavně využívání výhod objektově orientovaného programování. Tudíž při zápisu se objekt konvertuje do struktury a při čtení naopak. Popíšeme si, co se děje při zavolání metody pro čtení ze souboru HDF5. Adaptér obsahuje hlavní klíčovou třídu HDFWrapper, která obaluje veškerou funkcionalitu pro práci s knihovnou nižší vrstvy (adaptér obsahuje i jiné třídy, ale ty obsahují definice datových struktur nebo pomocnou funkcionalitu, kterou není nutno popisovat). Je využito strukturálního návrhového vzoru Adapter (Wrapper), kde nižší vrstva je v roli adaptovaného (Adaptee) a vyšší vrstvy, které volají HDF_Wrapper_CLI.dll jsou v roli klienta, viz obrázek 7.

Ještě než zavoláme metodu pro čtení, je třeba vytvořit instanci třídy HDFWrapper. Při vytvoření instance je potřeba zadat parametry, které dávají informace, kde je HDF5 soubor uložen, do jaké skupiny (group) chceme přistupovat v rámci HDF5 souboru a jaký chceme soubor dat (dataset). Konstruktor následně inicializuje nezbytně nutné proměnné a zjistí o jaký dataset se jedná, tedy zajistí si další metadata pro práci s HDF5 souborem. Zjistí, o jakou datovou strukturu se jedná, zda datový soubor obsahuje složené datové struktury (compound) nebo jednoduché datové struktury (elementy). Přesný popis jak pracovat se skupinami a jaký je rozdíl mezi složenou datovou strukturou a jednoduchou datovou strukturou naleznete v diplomové práci Martina Moudrého [36]. Poté co adaptér získá tato metadata, jsme připraveni volat metodu, která čte bloky dat (readBlock_Wrap). Na následujícím ukázkovém bloku kódu, viz výpis 8, můžeme vidět, že metoda vyžaduje parametr, v jakém pořadí chceme číst data, jestli chceme číst po řádcích nebo po sloupcích (při zadání hodnoty true, čteme po řádcích).

```
HDF_Wrapper_CLI::HDFObject^ HDF_Wrapper_CLI::HDFWrapper::readBlock_Wrap(bool
    readByRow, int fromCol, int toCol, int fromRow, int toRow) {
   columnSize = toCol - fromCol + 1; //x
   rowSize = toRow - fromRow + 1; //y
   obj = gcnew HDFObject(type, columnSize, rowSize);

   int size = columnSize * rowSize * info->getSize();
   structBuffer = new char[size];
   readBlock(readByRow, this->sfileName->c_str(), this->sgroupName->c_str(),
       this->sdatasetName->c_str(), compound, info, structBuffer, fromCol, toCol
       , fromRow, toRow);

   copyDataToManaged();

   return obj;
}
```

Výpis 8: Metoda adapteru pro čtení bloků dat z HDF5 souboru

Dále je potřeba zadat interval požadovaných dat, která chceme číst, tedy od jakého sloupce, po který sloupec včetně a u řádku platí to samé. Na ukázkovém bloku kódu si můžeme všimnout mírně odlišné syntaxe od běžného jazyka `C++`. Vzhledem k tomu, že třída HDFObject při své deklaraci obsahuje klíčové slovo ref, kterým říkáme, že třída bude patřit pod spravované `C++`, pak k spravovaným objektům přistupujeme skrze znak stříška (^). A tak tedy, že proměnná je typu HDFObject, tudíž vytvoření instance nemůžeme udělat přes klasické klíčové slovo new, ale je třeba použít gcnew (garbage collected new), čímž vytvoříme instanci pro spravovaný objekt. Zpět k průběhu metody čtení, vypočítáme si velikosti požadovaných počtů sloupců a řádků, vytvoříme instanci HDFObjektu, což je tzv. „super“ objekt, který obsahuje pole všech datových struktur, se kterými můžeme pracovat v rámci HDF5 abstrakce. Následně si alokujeme „buffer“, dle velikosti, kterou si uživatel žádal, a zavoláme metodu z nižší vrstvy. Poté konvertujeme získanou strukturu z „bufferu“ do objektu a výsledný objekt vracíme. Pokud je třeba přidat novou datovou strukturu v této vrstvě, je nutno definovat samotný objekt a jeho konstruktor. Následně je nutno implementovat konverzní metody, přidat pole do objektu HDFObject a přidat název struktury do výčtové třídy (enum class).

Nižší vrstva knihovny umožňuje také číst data pouze s jedním parametrem (readPointsParam_Wrap – čte parametr bodů, readBlockParam_Wrap – čte parametr bloků dat). Např. máme dopravní datovou strukturu a chceme pouze číst data parametru *CREATED_TIMESTAMP*, v tomto případě použijeme tuto metodu. Zde nastala komplikace při implementaci. Problém by byl snadno řešitelný v prostředí `C++/CLI`, ale vzhledem k tomu, že byla potřeba napsat stejnou

funkcionalitu v neřízeném C++, nabízelo se řešení, napsat funkcionalitu pro neřízené C++ a pak snadno tuto funkcionalitu portovat do prostředí řízeného C++/CLI. Vzniklý problém si popíšeme v následující podkapitole, která se bude zabývat implementací neřízeného C++ adaptéru. Jinak funkcionalita je podobná běžnému čtení bloků dat.

Adaptér obaluje metodu pro čtení bodů (readPoints_Wrap) v HDF5. Opět princip je podobný, ale parametry metody jsou jiné. Jediným vstupním parametrem je parametr „points“, což je spravované pole spravovaných datových typů String. Body jsou vždy uváděny v páru, kde první hodnota je sloupec a druhá hodnota je řádek jednoho bodu. Délka pole se proto dělí číslem 2, abychom získali počet bodů. Poté následuje cyklus, který převede spravované Stringy na nespravovaný datový typ unsigned long long. Poté je proces čtení stejný.

Adaptér obaluje i metodu pro vytvoření HDF5 souboru, skupin a datových souborů, ovšem proces je tak triviální, a proto nemá smysl jej zmiňovat. Obalená metoda pro přidání dat je jednoduchá, uvedeme si ukázku metody, viz výpis 9.

```
void HDF_Wrapper_CLI::HDFWrapper::append_Wrap(array<System::Object^>^ arr, int
    numOfCols, int numOfRows) {
   rowSize = numOfRows;
   columnSize = numOfCols;
   if (copyManagedToStructBuffer(arr) == false) {
      return;
   }

   append(this->sfileName->c_str(), this->sgroupName->c_str(), this->
       sdatasetName->c_str(), compound, info, structBuffer, numOfCols, numOfRows
       );
}
```

Výpis 9: Metoda adapteru pro zapisování bloků dat do HDF5 souboru

Parametrem je pole objektů a počet sloupců a řádků, neboť z délky pole nelze určit, jakou velikost sloupců a řádků má mít vkládané pole. Objekt se konvertuje do struktury, kde se mimochodem testuje, k jaké instanci daný objekt patří, abychom mohli identifikovat datovou strukturu, a poté se testuje, zda počet prvků pole souhlasí s počtem sloupců a řádků. Dále získáváme připravený „buffer“ a voláme metodu nižší vrstvy. Dostupná je obalená metoda (writeAtLoc_Wrap), která funguje stejně, ale nezapisuje na poslední volné místo v datovém souboru, ale na zadanou lokaci v datovém souboru, tímto je umožněno data přepisovat.

Formát HDF5 umožňuje u datových souborů pracovat s atributy. Každý datový soubor může mít libovolný počet atributů. HDF5 atribut je malý metadata objekt, který pomáhá blíže specifikovat konkrétní datový soubor. V atributu je navíc definována velikost dimenze. Např. dimenze 2x3 říká, že v atributu je uloženo 6 datových struktur. Tyto atributy můžeme číst a zapisovat. Čtení a zapisování atributů je téměř totožné se čtením a zapisováním datových bloků.

Při volání metody pro práci s atributy se definuje navíc pouze klíčový název atributu. Při čtení atributu je třeba zjistit metadata atributů. Pomocí nich zjistíme dimenzi, která nám určí, jak velký prostor potřebujeme alokovat pro příslušné objekty.

Jsou vytvořeny metody pro získání metadat ohledně HDF5 souboru, např. metoda pro zjištění všech objektů ve skupině (getGroupItems_Wrap), metoda pro zjištění všech atributů uložených v datovém souboru (getDatasetAttributes_Wrap), metoda pro zjištění dimenze jak datasetu (getDatasetDimension_Wrap), tak atributu (getAttributeDimension_Wrap). Naimplementováno je také jednoduché vyhledávání dle časového údaje. Vzhledem k povaze ukládaných dat a způsobu, jak se data budou ukládat, bylo rozhodnuto, že data budou mít časové razítko v určitých intervalech (např. záznam bude každou 1 minutu). Na začátku se zjistí ofset časových údajů, tento údaj zprostředkuje knihovna nižší vrstvy. Metoda, kterou se zjistí ofset je velmi triviální. Od časové hodnoty z druhého řádku se odečte časová hodnota z prvního řádku. Poté se hledaný časový údaj převede na formát Unix timestamp, který je použit pro uložená data v HDF5 a vypočítá se řádek, na kterém by se data měla nacházet. Vypočtený řádek se ověří vůči datům a vrátí se hodnota.

4.2 Adaptér pro `C++`

Popíšeme si knihovnu (HDF_Wrapper_CPP.dll) pro neřízené (Unmanaged) `C++`, která přistupuje do knihovny (TSdbHDF5.dll – označena jako nižší vrstva). Tato knihovna vznikla z důvodu kompatibility a podpory pod operačním systémem Linux (nutnost provozování knihoven na HPC clusteru). Jak bylo výše zmíněno, `C++/CLI` nemá podporu pod operačním systémem Linux jak pod platformou Mono, tak pod oficiálním běhovým prostředím .NET Core. Tento adaptér vykonává stejnou úlohu, jako předchozí adaptér `C++/CLI`. Ovšem má své omezení, vzhledem ke kompatibilitě a designu architektury, které se projeví na výkonu. Jedním z požadavků bylo, aby knihovna šla snadno importovat do jiných jazyků, jako je například Java, C# nebo statistický programovací jazyk R. Pro kompilátor Visual `C++`, který je používaný na Windows operačních systémech, je potřeba používat klíčové slovo „__declspec(dllexport)“, abychom dosáhli toho, že metody budou viditelné a přístupné pro aplikace, které budou využívat tuto knihovnu, tzv. je exportujeme z knihovny (na kompilátorech GCC a Intel pod operačním systémem Linux toto klíčové slovo není potřeba, metody jsou exportovány automaticky). Na ukázce zdrojového kódu, viz. výpis 10, můžeme vidět jednoduché makro „EXPORT“, které zjednodušuje proces exportování z pohledu kompatibility kompilátoru Visual `C++` vůči ostatním kompilátorům.

```
#if defined (_MSC_VER)
#include "..\HDF_header\TrafficInfo.h"
//...
#define EXPORT __declspec(dllexport)
#else
#define EXPORT
#include "TrafficInfo.h"
//...
#endif

namespace HDF_Wrapper {

        extern "C" EXPORT void readBlock_Wrap(const char* fileName, const char*
            groupName, const char* datasetName,bool readByRow, int fromCol,
           int toCol, int fromRow, int toRow, HDFObject **obj);
//...
```

Výpis 10: Ukázka makra EXPORT

Dále pro zjednodušení importování do ostatních programovacích jazyků bylo užito klíčového slova „extern "C"“ (exportování v C stylu), které usnadní volání jednotlivých funkcí knihovny (Nemusíme definovat vstupní bod - entry point. Stačí, aby se metoda jmenovala stejně jako volaná metoda). Při použití tohoto klíčového slova totiž nedochází k procesu, kterému se říká mangling (používá se český název komolení jmen, ovšem tento název se mi nelíbí) [39]. Mangling je záležitostí jazyka `C++` a úzce souvisí s polymorfismem. Když chceme zavolat metodu, která je obsažená v nějaké knihovně, vždy se hledá vstupní bod (entry point) dané metody. Daný vstupní bod musí být nějak identifikován. Metoda obsahuje identifikátor. Pokud máme knihovnu napsánu v jazyce C nebo je exportována v C stylu, pak její identifikátor je název dané metody. Pokud máme jazyk `C++`, dochází k manglingu, protože díky polymorfismu může být deklarována stejná metoda vícekrát s různými parametry, pak by docházelo k tomu, že identifikátor nebude unikátní (bude více identifikátorů se stejnými jmény). Abychom utvořili mangled metodu, vezme se její název a parametry a dle pravidel, které má definován kompilátor se převede do mangled tvaru, který je ve formě řetězce. Uvedeme si příklad mangled metody readBlock_Wrap: „`?readBlock_Wrap@HDF_Wrapper@@YAXPEBD00_NHHHHPEAPEAVHDFObject@1@@Z`“. Aplikační Binární Rozhraní (ABI), což je rozhraní mezi dvěma programovými moduly, které řeší, jak budou funkce volány a v jakém binárním formátu budou komunikovat. Bohužel nedefinuje žádný standard, podle jakých pravidel by měl být prováděn mangling funkcí. Takže v praxi je to tak, že v podstatě každý kompilátor provádí mangling podle svých pravidel. To je také jeden z důvodů, proč by se měl celý projekt ideálně kompilovat jedním typem kompilátoru. Přehledová tabulka,

Tabulka 3: Přehledová tabulka kompilátorů a ukázka manglingu metod [39]

Kompilátor	void h(int)	void h(int, char)	void h(void)
Intel C++ 8.0 for Linux	_Z1hi	_Z1hic	_Z1hv
HP aC++ A.05.55 IA-64			
IAR EWARM C++ 5.4 ARM			
GCC 3.x and 4.x			
IAR EWARM C++ 7.4 ARM	_Z<number>hi	_Z<number>hic	_Z<number>hv
GCC 2.9x	h__Fi	h__Fic	h__Fv
HP aC++ A.03.45 PA-RISC			
Microsoft Visual C++ v6-v10	?h@@YAXH@Z	?h@@YAXHD@Z	?h@@YAXXZ
Digital Mars C++			
Borland C++ v3.1	@h$qi	@h$qizc	@h$qv
OpenVMS C++ V6.5 (ARM mode)	H__XI	H__XIC	H__XV
OpenVMS C++ V6.5 (ANSI mode)		CXX$__7H__FIC26CDH77	CXX$__7H__FV2CB06E8
OpenVMS C++ X7.1 IA-64	CXX$_Z1HI2DSQ26A	CXX$_Z1HIC2NP3LI4	CXX$_Z1HV0BCA19V
SunPro CC	__1cBh6Fi_v_	__1cBh6Fic_v_	__1cBh6F_v_
Tru64 C++ V6.5 (ARM mode)	h__Xi	h__Xic	h__Xv
Tru64 C++ V6.5 (ANSI mode)	__7h__Fi	__7h__Fic	__7h__Fv
Watcom C++ 10.6	W?h$n(i)v	W?h$n(ia)v	W?h$n()v

viz tabulka 3, která vyobrazuje různá pravidla manglingu dle typu kompilátoru.

Je třeba zmínit i výkonnostní omezení, to se týká toho, že nelze importovat třídu HDFWrapper, alespoň pro jazyk C# [40]. Díky tomu si nelze držet instanci a tudíž metadata o HDF5. Proto bohužel při každém volání metody jsou znova a znova metadata zjišťována.

Popíšeme si problém, který vznikl při implementaci, který jsme si naznačovali v kapitole implementace adaptéru pro řízené `C++/CLI`, kdy jsme chtěli číst data pouze s jedním parametrem. Problém je ten, že neznáme datový typ požadovaného parametru při překladu. Datová struktura obsahuje různé datové typy, tudíž datový typ může být variabilní. Naštěstí tento problém řeší vlastnost `C++` a tou jsou šablony (templates). Problém je, že získáme požadovaná data v „bufferu“, ale neznáme jejich datový typ ani velikost. Vzhledem k tomu, že nižší vrstva poskytuje metodu pro zisk seznamu datových typů, které jsou odděleny středníkem, a známe pořadí požadovaného parametru, pak není problém zjistit si informace o požadovaném datovém typu, popřípadě pokud se jedná o pole, tak zjistit si jeho velikost. Samotná šablona nám pak převádí data z „bufferu“ přímo do řetězců.

```
class GeneralStruct
{
public:
    virtual void* GetBuffer() = 0;
    virtual std::string operator[] (int i) = 0;
    virtual int getSizeOf() = 0;
};

template <class T> class DataStruct : public GeneralStruct {
private:
```

```
    T* arr;
    int len;
public:
    DataStruct(int const count) { arr = new T[count]; len = 0; };
    DataStruct(int const count, int length) { arr = new T[count*length]; len = length; };
    ~DataStruct() { delete arr; };
    std::string operator[](int i) {
        std::stringstream val("");
        if (len == 0) {
            val << arr[i];
        }
        else {
            for (size_t l = 0; l < len; l++)
            {
                val << arr[l + (len*i)];
            }
        }
        return std::string(val.str());
    };
    void* GetBuffer() { return arr; };
    int getSizeOf() {
        if (len == 0) {
            return sizeof(arr[0]);
        }
        else {
            return len;
        }
    };
};
```

Výpis 11: Šablona v `C++` řešící konverzi variabilních datových typů

4.3 Adaptér pro C#

Krátce si popíšeme knihovnu (HDF_Wrapper_CS.dll), která přistupuje do knihovny (HDF_Wrapper_CPP.dll). Tato knihovna vznikla primárně pro potřeby provozování .NET prostředí pod operačním systémem Linux, kde programovací jazyk C# je plně podporován (Mono, .NET Core). Adaptér pro C# spolu s adaptérem pro neřízené `C++` dokážou plně nahradit adaptér `C++/CLI`, na jakékoliv platformě, ale je za to zaplacená daň ve formě malé výkonnostní penalizace (v předchozí podkapitole bylo zmíněno). Tato knihovna má jediný úkol a to alokovat a dealokovat ukazatele (pointery) pro objekty, které obsahují data struktur určená pro čtení nebo zápis. Knihovna je postavena na službě Platform Invoke (PInvoke), která umožňuje volat nespravovaný kód ze spravovaného prostředí.

4.4 HDF5 webová služba

Abychom rozšířili možnosti práce s HDF5 soubory a mohli poskytnout HDF5 jako službu, byla vyvinuta webová služba, která je postavena na Windows Communication Foundation (WCF) frameworku, čímž získáváme servisně orientovanou aplikaci. Webová služba má nastavené koncové body pro komunikaci se SOAP protokolem a REST API (výjimkou je pouze služba pro zápis, která má povolen pouze protokol SOAP). Téměř všechny servisní metody mají parametry jako řetězce, je to z důvodu použití REST API. Vždy při volání metody jsou řetězce převedeny na správné datové typy a testovány, zda obsahují validní hodnoty. V případě servisních metod pro zápis je přijímán jeden parametr a to objekt HDFWriter, který obsahuje všechny důležité členské proměnné pro zápis (název souboru, datový soubor, „superobjekt“, který obsahuje všechny datové struktury). Webová služba má nadefinováno ve svém konfiguračním souboru, kde je úložiště HDF5 souborů. Aby uživatel měl možnost zjistit, jaké HDF5 soubory jsou uloženy v úložišti, vznikla část služby pojmenovaná HDF.svc, která má metodu „GetDatastore“, která vrátí pole objektů typu HDFStore. Objekt HDFStore obsahuje členské proměnné pro identifikátor, název souboru, časová razítka vytvoření a upravení souboru. Dokumentace jak volat webovou službu a jak s ní pracovat, je uvedena v příloze, viz příloha C.

Chování webové služby je nastaveno tak, aby byla schopna obsloužit více požadavků ve stejném čase, čímž zvýšíme propustnost webové služby. U chování WCF služby se dají nastavit dva klíčové parametry, které nám dokážou zvýšit výkon dané WCF služby [41]. Jedním z parametrů je WCF instance (Instance Context Mode), která určuje, jak jsou objekty vytvořeny. Druhým parametrem WCF souběhem (Concurrency Mode) nastavujeme, kolik žádostí může být zpracováno objekty WCF. WCF instance má 3 hodnoty. První hodnota je „Per Call“, což znamená, že instance WCF služby, např. Read.svc je vytvořena při zavolání jakékoliv metody v této službě a po vrácení výsledku z metody je instance smazána. Druhou hodnotou je hodnota „Per Session“, kdy se vytvoří instance WCF služby pro konkrétní spojení (klient/server) při prvním zavolání metody, při dalším volání metody ve stejné WCF službě se využívá již vytvořené instance. Když klient dokončí všechny své požadavky, pak se instance smaže. Třetí hodnotou je hodnota „Single“, kde je vytvořena pouze jediná instance WCF služby pro všechny uživatele. WCF souběh má také 3 hodnoty. První hodnota je „Reentrant“, to znamená že, při vytvoření požadavku je vytvořeno jediné vlákno pro celou službu, které obsluhuje všechny požadavky a po dobu požadavku je WCF služba uzamčena. Druhou hodnotou je hodnota „Single“, kde požadavek blokuje jenom volání konkrétní metody ve WCF službě. Třetí hodnota je „Multiple“, kde každý požadavek je obsluhován jedním vláknem (je udržován fond vláken), tedy požadavky jsou zpracovávány paralelně. Je nutno si hlídat, aby nedocházelo k paralelnímu přístupu ke zdrojům (uzamykání proměnných apod.). Následující výpis kódu, viz výpis 12, popisuje, jak je nastaveno chování HDF5 webové služby.

```
...
    [ServiceContract]
```

```
    [ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Multiple,
        InstanceContextMode = InstanceContextMode.PerSession)]
    public class Read
    {
...
```

Výpis 12: Nastavení atributů ve WCF pro paralelní přístup

Nasazení webové služby je vcelku triviální. Na hostujícím serveru/počítači je třeba mít nainstalovánu Internetovou Informační službu (IIS) a nainstalovanou vlastnost „Aktivaci služby WCF protokolem HTTP“ (ideálně i jiným protokolem). Poté stačí nakopírovat patřičné soubory do složky, která je určená pro hostování webové služby. Měly by být nakopírovány tyto soubory: Global.asax, Web.config, Create.svc, Read.svc, Write.svc, HDF.svc a složka bin, která obsahuje knihovny: HDF_Wrapper_CLI.dll, hdf5.dll, HDFService.dll, szip.dll, TSdbHDF5.dll a zlib.dll. Poté je nutné nastavit správnou cestu k úložišti HDF5 souborů, toho dosáhneme úpravou souboru Web.config, kde v elementu appSettings u klíče HDFStore upravíme cestu k adresáři (webová aplikace musí mít nastavená plná přístupová oprávnění k adresáři). Nyní bychom měli mít funkční službu pro HDF5.

4.5 HDF5 klient pro webovou službu

Knihovna HDFWebClient.dll vznikla za cílem přistupovat k HDF5 souborům skrze webovou službu, kterou můžeme mít jako centrální úložiště všech HDF5 souborů. Tato knihovna je tedy protikusem k webové službě. Úkol knihovny je jednoduchý, volat službu skrze SOAP protokol a získána data opět převést na objekty. Pokud bychom chtěli změnit cílový server s HDF5 soubory, můžeme to buď provést skrze konfigurační soubor knihovny, nebo programátorskou cestou, kdy každá třída pro manipulaci s HDF5 soubory má naimplementovanou metodu setServer, kde jejím parametrem je adresa. Práce s knihovnou je stejná jako v případě knihovny HDF_Wrapper_CS.dll a HDF_Wrapper_CLI.dll.

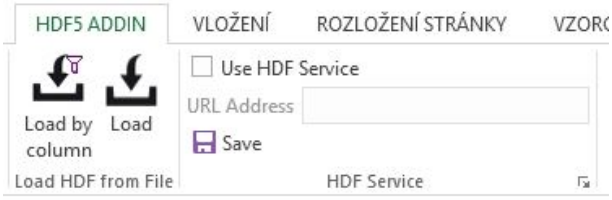
5 Implementace adaptéru pro čtení dat v rámci tabulkového procesoru MS Excel

Pro analýzu dat, statistiky a grafické znázornění, využijeme nejrozšířenější nástroj, tabulkový procesor Microsoft Excel. Způsoby, jak tvořit doplňky pro Excel, se nabízí hned 3. Omezením, které je důležité zmínit na začátku je, že všechny HDF5 knihovny jsou postaveny na 64 bitové platformě. Doplněk je proto nutné vyvíjet také pro 64 bitovou platformu, kvůli integraci knihoven (nelze volat z 32 bitového programu 64 bitovou knihovnu). Pro správnou funkcionalitu je nezbytné mít nainstalovanou sadu Microsoft Office pro 64 bitovou platformu. Jediný způsob, jak se vyhnout 64 bitové platformě, je zkompilovat HDF5 klienta pro webovou službu pro 32 bitovou platformu a omezit se pouze na práci s HDF5 soubory přes webovou službu.

První způsob, jak vyvíjet doplňky pro Microsoft Excel, je využití nástrojů pro Office 365. Software je zde poskytován jako služba (SaaS). Cloudová služba Office 365 je účtována formou předplatného. Jelikož Office 365 je postaven na webových technologiích, tak Microsoft nabízí JavaScript API pro vývoj doplňků [42]. Pro Excel využijeme rozhraní Excel JavaSript API (Excel.js), které má podporu Excel funkcí, práci s buňkami apod. Vyvinutý doplněk je nutné schválit v administrační části Office 365 pro organizaci. Výhodou je snadné nasazení doplňku. Nevýhodou může být složitější vývoj oproti tradičním metodám, jak vyvíjet doplňky pro Microsoft Office. Doplněk je vyvíjen pomocí HTML, JavaScript (jQuery) a CSS. Vzhledem k tomu, že budeme analyzovat velký objem dat, není Office 365 vhodnou volbou, neboť je výhodnější pracovat s daty lokálně z důvodu výkonu.

Druhý způsob vývoje doplňku pro Microsoft Excel je použití (maker) Visual Basic for Applications (VBA). VBA je programovací jazyk, který je rozšířením pro programovací jazyk Visual Basic. Oba tyto programovací jazyky používají stejnou běhovou knihovnu Visual Basic Runtime Library [43]. Oproti Visual Basicu je VBA rozšířeno o prvky, které umožňují snadno spolupracovat s produkty Microsoft Office, v případě Excelu můžeme v kódu pracovat s buňkami, listy, funkcemi atd. VBA nám umožňuje pracovat s Windows API a poskytuje možnost volání DLL knihoven. U Windows API bych se pozastavil, neboť možnost volání Windows API je velmi dobrou výhodou, ale také i velkým bezpečnostním rizikem. Tato možnost dala příležitost vzniku tzv. makro virům, což jsou viry napsané v jazyce VBA. Proto Microsoft Office poskytuje bezpečnostní úrovně pro spouštění maker. VBA kód se ukládá do samotného dokumentu, což v našem případě můžeme vnímat jako nevýhodu. Uživatel je fixován na nějakou šablonu, aby mohl pracovat s HDF5 soubory. Pro firemní používání má VBA jistá omezení, které se týkají bezpečnosti a nasazení [44]. Ovšem nespornou výhodou VBA je úzká integrace s Office aplikacemi, což přináší vyšší výkon, než poslední řešení, které si popíšeme. Programátorovi, který preferuje syntaxi jazyka Visual Basic a nepíše aplikaci pro firemní prostředí, lze toto řešení velmi doporučit.

Třetí možnost vývoje doplňku pro Microsoft Excel je sada nástrojů VSTO (Visual Studio Tools for Office), které jsou obsaženy v Microsoft Visual Studiu. Doplněk pomocí VSTO může být vyvíjen za pomocí programovacího jazyka Visual Basic, C# nebo F#. VSTO je podporováno



Obrázek 8: Náhled na kartu HDF5 doplňku v Excelu

od verze Microsoft Office 2003, kdy je umožněno hostovat .NET aplikace v rámci aplikací Office [45]. Původní záměr vývoje VSTO byl nahradit stávající VBA, ovšem zatím se to nepovedlo a VBA je často používaným řešením pro vývoj doplňků. Výhodou je, že VSTO kód je uložen mimo dokument nebo v sestavení (assembly), které je načteno se spuštěním aplikace [46]. Další výhodou oproti VBA je snadné použití .NET platformy, plná verze vývojového prostředí a snadná údržba kódu. Je to vhodné řešení pro firemní prostředí. VSTO je pomalejší oproti VBA, kvůli Interop vrstvě, která slouží jako most mezi spravovaným a nespravovaným kódem [47].

Z výše uvedených možností vývoje doplňku byla zvolena sada nástrojů VSTO. Základním stavebním prvkem pro vyvíjený doplněk je pás karet (Ribbon), kde si v designeru nadefinujeme prvky (tlačítka, popisky), kterými bude náš doplněk disponovat. Vývoj aplikace je pak stejný jako vývoj aplikace pro Windows Forms. Rozdílem ve vývoji oproti Windows Forms je používání Office knihoven, jmenných prostorů a tříd.

Na vyobrazeném obrázku, viz obrázek 8, můžeme na pásu karet vidět implementovaný HDF5 doplněk pro Excel. Doplněk lze používat, jak pro lokálně uložené soubory, tak pro webovou službu. Pokud chceme používat webovou službu, stačí pouze povolit možnost „Use HDF Service“ a vyplnit URL adresu k webovému serveru, kde je HDF5 webová služba provozována a stisknout tlačítko pro uložení. Po stisknutí tlačítka se ověří, zda je adresa validní a vytvoří se textový soubor (HDFExcel.txt) v umístění „`C:\Users\Uzivatel\AppData\Roaming`“, ve kterém je uložena konfigurace pro přístup k HDF5 webové službě. Výhodou ukládání konfiguračního souboru v tomto umístění je, že každý uživatel, který pracuje na stejném počítači, může mít svojí vlastní konfiguraci k HDF5 webovému úložišti.

Tlačítkem „Load“ nahrajeme obsah vybrané datové struktury. Pokud nemáme povoleno využívání HDF5 webové služby, při klepnutí na tlačítko „Load“ se otevře dialog, ve kterém vybereme HDF5 soubor z lokálního úložiště pro otevření. Pokud máme povoleno využívání HDF5 webové služby, pak vybíráme z dialogu HDF5 dostupných souborů na webovém serveru. Poté je scénář stejný, ať už máme či nemáme povoleno využití HDF5 webové služby. Vybereme si z nabídky datový soubor, ze kterého chceme získat data. Poté zadáme patřičné intervaly nebo jednotlivé body časových řad a záznamů, které chceme získat. Intervaly a body lze kombinovat, povolenými znaky jsou: znak pomlčky pro definování intervalu, znaky středník nebo čárka pro oddělení jednotlivých bodů nebo intervalů a číslice. Toto pravidlo je ověřováno regulárním výrazem po každém zadání znaku. Pokud vložená hodnota nevyhovuje pravidlu, je vyobrazen vykřičník u vkládaného pole. Poté jsou data vložena do aktivního sešitu. Počátečním bodem,

kde se mají vložit data je zvolena aktuální označená buňka v sešitu. Data jsou vypisována ve formátu, kdy jsou časové řady řazeny pod sebou.

V případě tlačítka „Load by column“ se nám nahraje pouze uživatelem daný atribut z datové struktury. Průběh procesu je stejný jako u tlačítka „Load“. Rozdíl je pouze u dialogu, kdy uživatel zadává intervaly a body, kde ještě navíc vybere z nabídky atribut z datové struktury, který chce získat.

Instalace doplňku je jednoduchá, stačí spustit instalační skript, který zajistí zkopírování knihoven do složky „`C:\Windows`“ a spustí instalační balíček HDF5 doplňku pro Excel. Odinstalování balíčku má dvě části, spustí se skript pro odinstalaci, který zajistí smazání knihoven a smazání pomocných souborů. Následně je potřeba odinstalovat doplněk pomocí Programů a funkcí v Ovládacím panelu, kde je položka „HDF5 Excel AddIn“.

Popíšeme si 2 různé způsoby, jak zapisovat data do buněk v Excelu z programového kódu [48]. První způsob, kterým lze zapisovat data, je pracovat s buňkami jako s dvourozměrným polem. Pro programátora, který píše doplněk pro Excel a implementovaný doplněk pracuje s buňkami, pak tento způsob práce s buňkami je svádějící. Objektový model buněk v Excelu nabízí různé přístupy k buňkám a každý způsob přístupu má své opodstatnění. Tedy v případě, že zapisujeme velký objem dat, iterovat buňku po buňce je časově náročné a tento způsob není doporučen. Rychlost tohoto způsobu přístupu lze zvýšit tím, že se při zápisu dat vypne dočasně aktualizace buněk, což znamená, že uživatel neuvidí průběh zápisu do buněk. Avšak zrychlení není markantní. Druhý způsob je výrazně rychlejší a je doporučen. Buňkami se neiteruje. Zvolí se první buňka a poslední buňka (na hlavní diagonále) a zavolá se metoda pro získání rozsahu, která vyžaduje právě zvolené buňky. Objekt určený pro ukládání rozsahu obsahuje atribut „Value“, který vyžaduje objekt, který obsahuje hodnoty, které se mají vložit. V našem případě vkládáme objekt, který je reprezentován jako dvourozměrné pole a obsahuje data, která chceme vložit.

6 Příprava dat pro experimenty a provedení experimentů

Tato kapitola se zabývá popisem generátoru dat, měřením rychlosti HDF5 adaptérů, kde porovnává adaptér pro řízené `C++/CLI` a adaptér pro neřízené `C++` v kombinaci s adaptérem pro C#. Provedeme také experiment na výpočetním uzlu superpočítače Salomon, kde je potřeba testovat paralelní přístup k HDF5 přes vytvořený adaptér. Nakonec byl spuštěn zátěžový test WCF webové služby.

Experimenty byly prováděny na dvou různých hardwarových zařízeních. Jedním z nich je jeden výpočetní uzel superpočítače Salomon, jehož konfigurace je následující [49]: dva dvanáctijádrové procesory Intel Xeon E5-2680v3 (2,5GHz), 128GB RAM a bylo pracováno s adresářem home, jehož diskové úložiště má propustnost 6GB/s. Druhý zařízením je notebook HP ProBook 6550b, který má tuto konfiguraci: dvoujádrový procesor Intel Core i5-450M (2,6GHz), 4GB RAM a SSD disk ADATA SX900, jehož propustnost pro zápis je 540MB/s a pro čtení 560MB/s (IOPS je 91 000). Zařízení byla volena dle závislostí testované platformy. Jakákoliv část HDF5 architektury, která je závislá na knihovně C++/CLI, musí být provozována na operačním systému Windows (proč tomu tak je, bylo vysvětleno v kapitole o řízeném adaptéru), tudíž je v tomto případě zvolena konfigurace notebooku. Pokud lze platformu testovat pod operačním systémem Linux, byl zvolen superpočítač.

Při testech bude používána navíc datová struktura Weather (počasí), která má následující atributy: atribut *TIMESTAMP* pro ukládání časového razítka, atribut *NAME* pro uložení názvu měřící stanice, atribut *DEGCEL*, který obsahuje hodnotu stupňů v jednotkách Celsia. Atribut *DEGFAH* obsahuje hodnotu v jednotkách Fahrenheita. A poslední atributy *LON* a *LAT*, které obsahují GPS souřadnice měřící stanice.

6.1 Generátor dat

Pro potřeby testování, vývoje adaptérů a vytvoření architektury bylo potřeba mít data, se kterými bychom mohli pracovat. Proto vznikla knihovna pro generování dat, která byla vyvinuta v jazyce `C++`. Generátor generuje data pro datové struktury Traffic (Here) a Weather. Bylo potřeba vytvořit generátor, který by vytvořil velký objem dat v malém časovém měřítku. Toho bylo dosaženo díky paralelizaci. Pro generování dat je potřeba CSV soubor. CSV soubor byl získán v rámci Finských OpenData (InfoTripla). Ten obsahuje informace o měřících stanicích, jako např. název stanice, souřadnice stanice, úroveň silnice, kde je měřící stanice umístěna apod. Metoda pro generování dat vyžaduje tři parametry, časové razítko, od jaké časové hodnoty chceme data generovat. Dalším parametrem je časový krok a poslední parametr je počet segmentů (hodnot), které chceme získat. Při vykonávání metody pro generování se načtou měřící stanice, inicializuje se vektor pro generované objekty a inicializuje se vektor pro vlákna, tzv. „thread pool“, který je následně naplněn vlákny, které již začaly plnit svůj úkol.

Pro dopravní data (Traffic) jsou dva vektory pro vlákna, neboť každý směr silnice je generován jedním vláknem. Jsou použity dva zdroje generátorů. Jedním z nich je klasický „rand“,

který je použit na generování všech atributů, kromě generování rychlosti. Většina atributů se generuje na základě definovaných pravidel, aby hodnoty byly pokud možno věrohodné s reálnými hodnotami datové struktury, kterou používá Here. Pro rychlost je generování složitější. Pro věrohodnost je zvolen generátor „mt19937“, který má lepší statistické chování, než generátor „rand“ [50]. Cílem generování rychlosti je generovat hodnotu, která se mění plynule a dokáže simulovat např. dopravní zácpu či kolizi. Důvod pro takové generování dat je prostý, uložená data v HDF5 souboru je možné analyzovat a tudíž jsou velice věrohodná k reálným datům. Dle načtených informací o měřících stanicích je identifikováno, jaké rychlostní data se budou generovat. Zda půjde o dálniční komunikaci, rychlostní komunikaci nebo komunikaci obecní. V případě dálnice je vrchní hranice nastavena na 133km/h a spodní hranice 30km/h (v případě dopravní zácpy hodnota nebude nižší). Rychlostní komunikace má vrchní hranici nastavenu na 93km/h a spodní hranici na 20km/h. Obecní komunikace má vrchní hranici 55km/h a spodní hranici 10km/h. Generátor „mt19937“ umožňuje nastavit pravděpodobnostní rozdělení, kde jsme použili normální rozdělení [51], abychom získali vyvážený generátor, ale s náhodným přírůstkem. Pro definici normálního rozdělení jsou potřeba dva parametry. Průměr, který je pro všechny typy komunikací stejný a směrodatná odchylka, která je pro typ komunikace odlišná. Díky správnému nastavení normálního rozdělení můžeme dosáhnout simulace dopravní zácpy. Poté je vygenerovaná hodnota přičítána nebo odčítána od rychlosti v závislosti na rychlostních hranicích.

Generování dat pro počasí je obdobné jako u dat dopravních. Místo dopravních hranic jsou definované hranice časové, tedy od 6 hodiny do 20 hodiny je teplota vyšší, než od 21 hodiny do 5 hodiny ráno. Zde generování není tak propracováno, že by bylo detekováno, o jaké roční období se jedná. Tudíž denní hodnota může kolísat od 10 do 32 stupňů Celsia a noční od 5 do 17 stupňů Celsia. I zde se jedná o plynulý vývoj teploty, kde pro generování přírůstku byl zvolen generátor „mt19937“ s normálním rozdělením.

Vzhledem k tomu, že generátor byl vyvinut s důrazem na paralelizaci, bylo potřeba řešit uzamykání proměnných a obecně práci s vlákny. Generátor po vygenerování dat vracel aplikacím vektor, který obsahoval řetězce. Generátor disponuje podpůrnými metodami, jako je např. seznam datových typů generované struktury, získání časového razítka, kdy byla data vygenerována pro potřeby pojmenování datového souboru, vysvětlivky jednotlivých atributů atd.

6.2 Porovnání výkonu mezi adaptéry

V této podkapitole budeme porovnávat rychlost zpracování adaptérů. Porovnáme adaptér pro řízené `C++`/CLI proti kombinaci adaptérů pro C# a pro `C++`. Pro testování byl vytvořen projekt „HDFCompareAdapters“, který je naprogramován v jazyce C#. Testovací metodika je následující. Vytvoří se totožné 2 HDF5 soubory. Jeden skrze řízený `C++`/CLI adaptér a druhý skrze C# adaptér, který následně volá `C++` adaptér. Testovací program má nastaveno testování pro 2000 řádků a 2000 sloupců, tudíž budeme pracovat se 4 milióny datových objektů. Načte se vygenerovaný soubor, který obsahuje data, které použijeme pro zápis. Z načtených dat se vytvoří objekt, který je použit jako parametr metody pro zápis. Poté se změří čas, jak dlouho trvalo

Tabulka 4: Výsledky porovnání výkonů adaptérů pro datovou strukturu Traffic

Traffic (jednotky: sekundy)		append	readPointsParam	readBlock
C# + C++	Průměr	1,83	52,72	2,83
	Maximum	1,9	52,9	2,94
	Minimum	1,77	52,6	2,74
C++/CLI	Průměr	4,01	51,71	2,02
	Maximum	4,15	51,9	2,06
	Minimum	3,89	51,52	1,99

Tabulka 5: Výsledky porovnání výkonů adaptérů pro datovou strukturu Weather

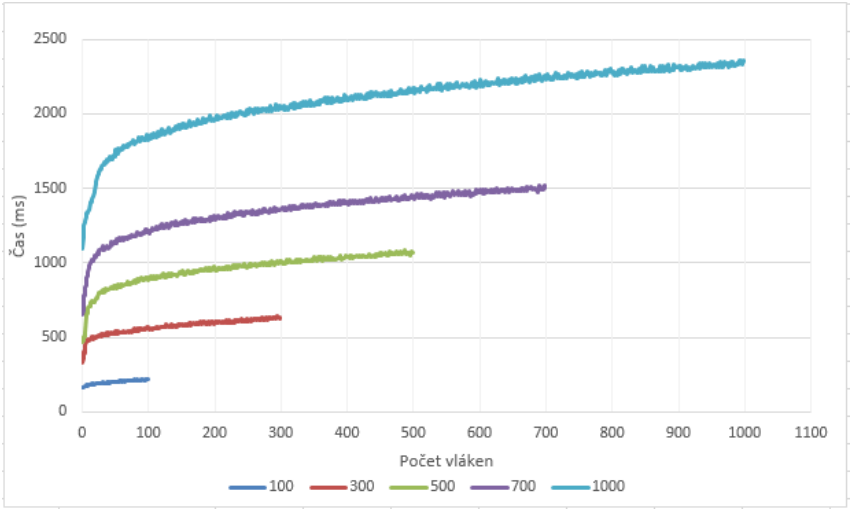
Weather (jednotky: sekundy)		append	readPointsParam	readBlock
C# + C++	Průměr	9,94	50,05	6,16
	Maximum	10,14	50,39	6,2
	Minimum	9,77	49,75	6,1
C++/CLI	Průměr	11,29	50	5,71
	Maximum	11,33	50,2	6,17
	Minimum	11,26	49,79	5,47

volání metody pro zápis. Dalším aspektem je testování náhodného přístupu do HDF5 souboru. Vygenerují se 4 milióny bodů, které budeme číst. Opět jsou volány metody pro čtení jednotlivých bodů, ale byla zvolena verze, která čte daný atribut z datové struktury. Metoda pro čtení bodů daného atributu byla zvolena záměrně, jelikož je implementačně složitější (metoda staví na stejných základech, jako metoda pro čtení náhodných bodů celých datových struktur, ale pracuje navíc s generickými datovými typy), oproti metodě, která čte náhodné body celé datové struktury. Poslední měřenou metodou je metoda pro čtení bloků dat, zde konkrétně čteme najednou sekvenčně celý HDF5 soubor. Test byl proveden na notebooku. C++/CLI a C++ adaptéry byly kompilovány s optimalizací O2 (optimalizace rychlosti).

Test byl proveden 3krát a jednotlivé výsledky testu byly zprůměrovány. V tabulce 4 můžeme vidět výsledky pro datovou strukturu Traffic. V tabulce 5 jsou zobrazeny výsledky pro datovou strukturu Weather. Testování obou datových struktur dopadlo podobně. Pro zápis byla rychlejší kombinace adaptérů C# a C++. Pro operace čtení, ať už náhodných bodů nebo čtení bloků, vítězil adaptér C++/CLI. Vysvětlíme si, proč má řízený adaptér horší výsledky při zapisování. Jde o dva výkonnostní problémy. Jeden z nich nastává, když se konvertují objekty do struktur. Jedno zpomalení způsobuje detekce typu struktury, která je jinak implementována, než v neřízeném adaptéru. U řízeného adaptéru je detekce prováděna automaticky, kdežto u neřízeného adaptéru musí informaci o datovém typu dodat programátor. Druhý výkonnostní problém, který je závažnější pro výkon, než předešlý, je konverze řetězců ze spravované části do nespravované části adaptéru C++/CLI. Obecně by měla být pomalejší kombinace adaptérů, neboť pracujeme se dvěma vrstvami, namísto jedné.

Tabulka 6: Výsledky paralelního přístupu skrze `C++` adaptér

Počet vláken (jednotky: milisekundy)	100	300	500	700	1000
Průměr	199,76	572,89	958,548	1350,25	2105,78
Maximum	220,30	645,43	1083,11	1517,96	2356,87
Minimum	123,50	307,07	423,66	656,11	932,77



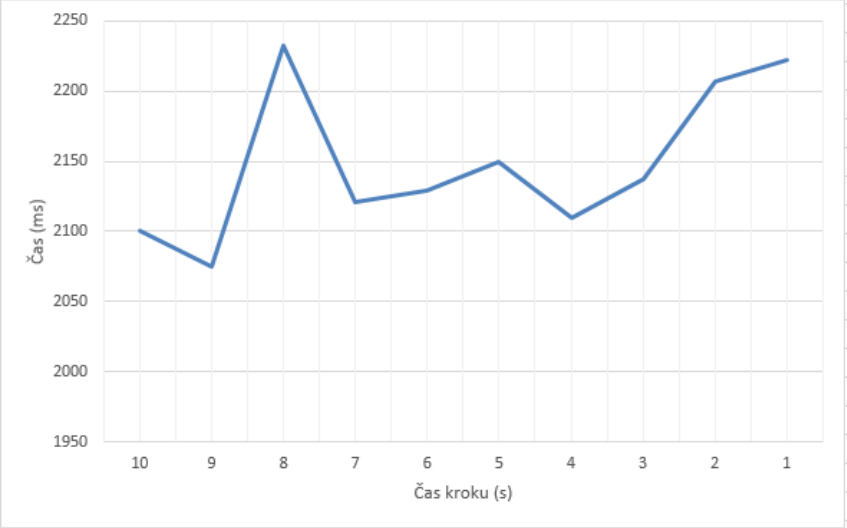
Obrázek 9: Graf znázorňující závislost času zpracování požadavku na počtu vláken

6.3 Test paralelního přístupu

V této podkapitole budeme testovat a měřit paralelní přístup k HDF5 souboru. Abychom mohli pracovat s HDF5 soubory paralelně, musí být HDF5 knihovna (hdf5.dll) zkompilovaná s příznakem „threadsafe“ [52]. Pro testování HDF5 souborů byl vytvořen projekt „HDFPerformanceTester“, který byl vyvinut v programovacím jazyce C++. Test byl proveden na výpočetním uzlu superpočítače Salomon podle následující testovací metodiky. Při spuštění testu byl vytvořen určitý počet vláken v jednom okamžiku. Každé vlákno četlo určitý blok dat (HDF5 soubor byl pomocí vláken přečten celý) o velikosti 10 sloupců a 100 řádků, přičemž se měřila doba vykonání metody pro čtení bloku. Poté byl výsledek zapsán na disk. Zde můžeme vidět tabulku, viz tabulka 6, výsledků pro určité počty vláken. Důležitá hodnota je maximum, která nám říká, jaká byla nejdelší doba pro zpracování požadavku. Graf, viz obrázek 9 nám znázorňuje závislost času zpracování požadavku na počtu vláken. Každá vynesená křivka představuje počet vytvořených vláken. Křivka je sestavená z bodů, kde jeden bod je ekvivalentní jednomu vláknu. Když si stanovíme pomyslnou hranici 1,5 sekund (s patřičnou tolerancí), skrze kterou nepřekročíme mez, pak zjistíme, že 700 vláken je optimálních pro splnění této podmínky. Při potřebě zpracování vyššího počtu vláken je dále nutné úlohu optimalizovat nebo adaptér provozovat na výkonnějším hardwaru. Je nutno upozornit na omezení při zápisu. I když zkompilujeme knihovnu jako paralelní, nelze do HDF5 souboru zapisovat paralelně, paralelně lze jen číst, viz. výše.

Tabulka 7: Výsledky zátěžového testu `C++` adaptéru

Počet vláken (jednotky: milisekundy)	100	300	500	700	1000
Průměr	199,76	572,89	958,548	1350,25	2105,78
Maximum	220,30	645,43	1083,11	1517,96	2356,87
Minimum	123,50	307,07	423,66	656,11	932,77



Obrázek 10: Graf znázorňující průměrný čas vykonání metody při zátěžovém testu

Jedním z testů paralelního přístupu byl zátěžový test, který měřil čas zpracování metody pro čtení bloků dat z HDF5 souboru se stejnou metodikou testování jako předchozí test (10 sloupců, 100 řádků). Byl spuštěn odpočet času od 10 do 1 vteřiny, kde v každé jedné vteřině bylo spuštěno 1000 požadavků, chceme-li 1000 vláken. Důležitá je hodnota maximum, která nám značí nejdelší čas daného požadavku, viz tabulka 7. Pro znázornění je průměrná hodnota z tabulky vynesena do grafu, viz obrázek 10.

6.4 Zátěžový test HDF5 webové služby

V této podkapitole bude testována HDF5 webová služba, která je postavena na technologii .NET WCF. Test byl proveden na hardwarové konfiguraci notebooku. Pro provedení zátěžového testu byl zvolen software Apache JMeter. Při testování zátěže bylo pro porovnání využito škálování výkonu IIS serveru, kde je možnost zvýšit tzv. worker (pracovní) procesy (WP). Worker proces je proces (w3wp.exe), který provozuje webovou aplikaci, v našem případě HDF5 webovou službu, a je zodpovědný za zpracování požadavků, které jsou zaslány na webový server pro specifickou aplikaci [53]. Jelikož tento test byl proveden v prostředí operačního systému Windows, bylo potřeba si zkompilovat knihovnu HDF5 s parametrem „threadsafe“ [52]. Je důležité podotknout, že parametr „threadsafe“ není oficiálně podporován pod platformou Windows a není doporučeno toto sestavení používat pro produkční účely. Testovací metodika byla následující.

Tabulka 8: Výsledky zátěžového testu HDF5 webové služby pro metodu GetBlockByColumn

	Počet vláken/WP (jednotky: milisekundy)			
	50/1	100/1	50/4	100/4
Průměr	10,18	33716,37	8,24	1061,47
Maximum	56	115412	29	2012
Minimum	5	23	5	34

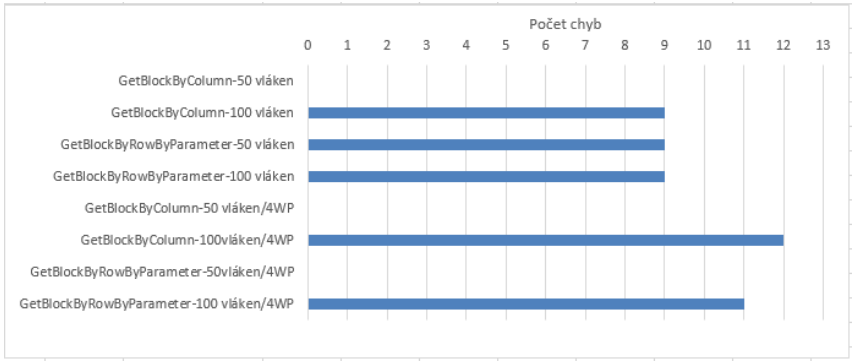
Tabulka 9: Výsledky zátěžového testu HDF5 webové služby pro metodu GetBlockByRowByParameter

	Počet vláken/WP (jednotky: milisekundy)			
	50/1	100/1	50/4	100/4
Průměr	23460,17	86018,88	20,48	846,82
Maximum	70659	182451	170	3457
Minimum	385	54	10	22

V softwaru Apache JMeter byl vytvořen element „Thread Group“, který určoval, kolik vláken bude použito pro zátěžový test. Byla zvolena varianta 50 a 100 vláken. Dále byl zvolen element „Random Controller“, který náhodně vybírá z množiny elementů „HTTP Request“. V množině je definováno 10 elementů „HTTP Request“, kde jsou náhodně definované požadavky pro metodu „GetBlockByColumn“ nad datovou strukturou Traffic. Velikost dotazované datové struktury je vždy stejně velká, 10 sloupců a 10 řádků. Dále byl využit element „XML Assertion“, abychom zjistili, zda získané XML z webové služby je validní. Dále byla zvolena varianta škálování výkonu s výchozím 1 worker procesem a varianta s navýšením na 4 worker procesy. Stejná testovací metodika byla zvolena pro otestování metody „GetBlockByRowByParameter“.

Dle výsledných tabulek, viz tabulky 8, 9, a grafu, viz obrázek 11, znázorňujícího míru chybovosti můžeme usoudit, že navýšení worker procesů (WP) má smysl, neboť získáváme lepší čas zpracování volané metody, ale míra chybovosti je dle mého názoru vysoká. Chybný požadavek byl vyhodnocen v případě, že nebylo navráceno validní XML, žádné XML nebo čas požadavku byl delší, než jedna minuta. Parametr „threadsafe“ není oficiálně podporován pod platformou Windows a není doporučeno toto sestavení používat pro produkční účely. To je pravděpodobně důvodem vysoké míry chybovosti. Při sestavení knihovny bez parametru „threadsafe“ se žádná chybovost neprojevila. Používání knihovny bez parametru „threadsafe“ pro webovou službu je nevhodné, neboť je velká pravděpodobnost, že při dvou a více současných požadavcích nebude uspokojen ani jeden z nich. Je proto lepší předpokládat chybovost a v případě chyby řešit chybně vyhodnocený požadavek opakovaně.

Jedna z možností, jak uskutečnit tento test byla, že místo notebooku by byly použity výpočetní uzly Salomon jako úložiště pro HDF5 soubory a IIS server (případně IIS farma), který by sloužil uživatelům pro přístup k HDF5 souborům. Ovšem uskutečnění zátěžového testu webové služby není nutné při této zamýšlené konfiguraci, protože bychom dosáhli jiné míry chybovosti. Dále bychom testovali propustnost sítě a ne odolnost zátěže webové služby. Získali bychom



Obrázek 11: Graf znázorňující míru chyb při testování zátěže HDF5 webové služby

zkreslené výsledky, které by ovlivňovala právě zmíněná propustnost sítě.

7 Závěr

Cílem této práce bylo realizovat rešerši OpenData zaměřených na dopravu a také vytvořit rešerši komerčních zdrojů, taktéž zaměřených na dopravní informace. Krom nalezených zdrojů bylo nutné analyzovat i zdroje, které byly zadány v rámci zadání diplomové práce a zjistit, zda jsou zdroje (Inrix, Google) vhodné pro poskytování dat. Nalezené a zadané zdroje bylo potřeba analyzovat, zjistit vlastnosti zdrojů a stručně popsat podmínky poskytování dat. Dalším z cílů bylo vybrat alespoň jeden zdroj z nalezených zdrojů dat, což se povedlo.

Dalším cílem bylo navázat na diplomovou práci Martina Moudrého, která se zabývá rozšířením HDF5 o ukládání časových řad [36]. Zde bylo motivací najít vhodný způsob ukládání časových řad, tedy dopravních dat a jiných datových struktur, které obsahují časovou složku. Cílem bylo vyvinout adaptér nad vzniklou knihovnou Martina Moudrého. Podmínkou adaptéru bylo, aby byl multiplatformní, tvořil abstrakci nad HDF5 formátem a v případě změny formátu ukládání dat byl snadno upravitelný pro nový formát. V případě změny formátu, díky tomuto konceptu, není třeba upravovat již další vytvořené vrstvy (architekturu) vzniklé nad vytvořeným adaptérem. Účelem implementace bylo potřeba vytvořit doplněk pro Excel, který by četl data uložená v HDF5 souboru. Dále bylo záměrem vytvořit webovou službu, která umožňuje práci s HDF5 souborem a vytvoření knihovny využívající webovou službu. Všechny zadané cíle byly úspěšně splněny. Následně bylo třeba provést experimenty nad vzniklou architekturou, které se zabývaly testováním výkonu a porovnávání výkonu mezi vzniklými adaptéry. Zde byly výsledky testů pozitivní, ale v případě testování webové služby i negativní, kde jsme zaznamenali značnou chybovost výsledků, která ale mohla být způsobena používáním nedoporučené konfigurace HDF5 knihovny. V implementaci adaptérů je dostatečný prostor pro optimalizaci adaptérů, z hlediska paralelizace, popřípadě využití OpenMP API.

Petr Zubek

Literatura

[1] *Open data stories | Open Data Institute* [online]. 65 Clifton Street, London, 2015 [cit. 2015-09-15]. Dostupné z: https://theodi.org/stories

[2] *Open Knowledge: What is Open?* [online]. [cit. 2015-09-15]. Dostupné z: https://okfn.org/opendata/

[3] *Otevřená data. Ministerstvo vnitra České republiky* [online]. 2015 [cit. 2015-09-15]. Dostupné z: http://www.mvcr.cz/clanek/otevrena-data.aspx?q=Y2hudW09Mg

[4] *5-star Open Data* [online]. 2012 [cit. 2015-09-15]. Dostupné z: http://5stardata.info/en/

[5] *Google API Documentation* [online]. [cit. 2016-01-28]. Dostupné z: https://developers.google.com/products/

[6] *RFC 5545 - Internet Calendaring and Scheduling Core Object Specification (iCalendar)* [online]. 2009 [cit. 2016-01-28]. Dostupné z: https://tools.ietf.org/html/rfc5545#section-3.8.5

[7] RODRIGUEZ, Alex. *RESTful Web services: The basics* [online]. 2008 [cit. 2016-04-09]. Dostupné z: https://www.ibm.com/developerworks/webservices/library/ws-restful/

[8] *Google Maps Web Service APIs|Google Developers* [online]. [cit. 2016-01-28]. Dostupné z: https://developers.google.com/maps/web-services/

[9] *Proč je minifikace JavaScriptu dobrá pro váš web* [online]. [cit. 2016-01-28]. Dostupné z: http://mareksommer.cz/blog/proc-minifikovat-javascript.html

[10] *Pricing and Plans|Google Maps APIs|Google Developers* [online]. [cit. 2016-01-28]. Dostupné z: https://developers.google.com/maps/pricing-and-plans/#details

[11] *Google APIs Terms of Service|Google Developers* [online]. [cit. 2016-01-28]. Dostupné z: https://developers.google.com/terms/

[12] *Kapitola III. ITS (Inteligentní Dopravní Systémy) (ČÁST 2): Zvýšení vědeckovýzkumného potenciálu pracovníků a studentů technických vysokých škol v oblasti dopravy* [online]. Ostrava, 2009 [cit. 2015-10-25]. Dostupné z: http://projekt150.ha-vel.cz/node/93

[13] HRUBEŠ, Pavel. *Studie „Zmapování služeb a dat v oblasti FCD (Floating Car Data) pro využití v rámci informačních systémů ŘSD“* [online]. In: . Praha, 2010 [cit. 2016-04-09]. Dostupné z: http://www.lss.fd.cvut.cz/Members/langr/uaemka/fcd/at_download/file

[14] HAYWARD, Mike. *Technologies and Techniques for Collecting Floating Car Data* [online]. [cit. 2015-09-18]. Dostupné z: http://cordis.europa.eu/pub/telematics/docs/tap_transport/iuttfwsa.pdf

[15] *InfoTripla* [online]. [cit. 2016-02-25]. Dostupné z: http://tie.digitraffic.fi/sujuvuus/ws/lamData

[16] *Datex2 - Deployments* [online]. [cit. 2015-10-25]. Dostupné z: http://www.datex2.eu/datex-node

[17] *DATEX informasjonsside / Statens vegvesen* [online]. 2015 [cit. 2016-01-28]. Dostupné z: http://www.vegvesen.no/om+statens+vegvesen/om+organisasjonen/For+utviklere+API/Datex?lang=nn

[18] *Datex / Statens vegvesen* [online]. 2015 [cit. 2016-01-28]. Dostupné z: http://www.vegvesen.no/en/Traffic/On+the+road/Datex2

[19] *Creative Commons Attribution 4.0 International CC BY 4.0* [online]. [cit. 2016-01-28]. Dostupné z: https://creativecommons.org/licenses/by/4.0/

[20] *Real time traffic information - Trafikverket* [online]. 2014 [cit. 2015-10-25]. Dostupné z: http://www.trafikverket.se/en/startpage/operations/Operations-road/Traffic-information/Real-time-traffic-information/

[21] *Traffic: TomTom vs Waze vs Garmin (in the UK) / TomTom EN* [online]. 2015 [cit. 2016-01-28]. Dostupné z: https://en.discussions.tomtom.com/tomtom-traffic-live-and-other-services-47/traffic-tomtom-vs-waze-vs-garmin-the-uk-982275

[22] *Transport Protocol Experts Group (TPEG) - TISA* [online]. [cit. 2016-01-28]. Dostupné z: http://tisa.org/technologies/tpeg/

[23] *Home - INRIX* [online]. 2016 [cit. 2016-01-28]. Dostupné z: http://inrix.com/

[24] *TomTom Real Time Traffic – The most accurate Information* [online]. 2015 [cit. 2016-01-28]. Dostupné z: https://www.tomtom.com/en_gb/licensing/products/traffic/real-time-traffic/

[25] *Here (company)* [online]. [cit. 2016-01-28]. Dostupné z: https://en.wikipedia.org/wiki/Here_(company)

[26] *Traffic API XML* [online]. [cit. 2016-01-28]. Dostupné z: http://traffic.cit.api.here.com/traffic/6.0/xsd/flow.xsd?app_id=DemoAppId01082013GAL

[27] *Traffic API Developer's Guide: Version 6.0.28.1* [online]. In: . [cit. 2016-01-28]. Dostupné z: https://developer.here.com/documentation/download/traffic_nlp/6.0.28.1/Traffic

[28] *Reading traffic flow data from HERE maps rest API - Stack Overflow* [online]. [cit. 2016-01-28]. Dostupné z: http://stackoverflow.com/questions/28476762/reading-traffic-flow-data-from-here-maps-rest-api

[29] *Here api - Traffic flow data XML tags meaning - Stack Overflow* [online]. [cit. 2016-01-28]. Dostupné z: http://stackoverflow.com/questions/22394499/traffic-flow-data-xml-tags-meaning

[30] *Terms and Conditions - HERE Developer* [online]. [cit. 2016-01-28]. Dostupné z: https://developer.here.com/terms-and-conditions

[31] *Pricing and Plans - HERE Developer* [online]. [cit. 2016-01-28]. Dostupné z: https://developer.here.com/plans/api/consumer-mapping

[32] *TPEG Services - TISA* [online]. [cit. 2016-01-28]. Dostupné z: http://tisa.org/technologies/tpegtm-services-map/

[33] *RFC 3339 - Date and Time on the Internet: Timestamps* [online]. 2002 [cit. 2016-01-28]. Dostupné z: https://tools.ietf.org/html/rfc3339

[34] *Key Concepts - Venue Maps API - HERE Developer - HERE Developer* [online]. [cit. 2016-01-28]. Dostupné z: https://developer.here.com/rest-apis/documentation/venue-maps/topics/key-concepts.html

[35] *Adapter (wrapper)* [online]. [cit. 2016-02-22]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/adapter-wrapper-navrhovy-vzor

[36] MOUDRÝ, Martin. *Rozšíření HDF5 o ukládání časových řad* Ostrava, 2016. Diplomová práce. VŠB-TUO, FEI. Vedoucí práce Ing. Jan Martinovič, Ph.D.

[37] *What is HDF5?* [online]. 2011 [cit. 2016-03-2]. Dostupné z: https://www.hdfgroup.org/HDF5/whatishdf5.html

[38] *HDF5 Technologies* [online]. 2011 [cit. 2016-03-2]. Dostupné z: https://www.hdfgroup.org/about/hdf_technologies.html

[39] *Name mangling* [online]. [cit. 2016-03-28]. Dostupné z: https://en.wikipedia.org/wiki/Name_mangling

[40] *How do I DllExport a C++ Class for use in a C# Application - Stack Overflow* [online]. 2011 [cit. 2016-03-31]. Dostupné z: http://stackoverflow.com/questions/4741035/how-do-i-dllexport-a-c-class-for-use-in-a-c-sharp-application

[41] *WCF Concurrency (Single, Multiple, and Reentrant) and Throttling - Code Project* [online]. 2013 [cit. 2016-03-31]. Dostupné z: http://www.codeproject.com/Articles/89858/WCF-Concurrency-Single-Multiple-and-Reentrant-and

[42] *Office Dev Center - Docs - Excel JavaScript API programming overview* [online]. [cit. 2016-04-05]. Dostupné z: https://dev.office.com/docs/add-ins/excel/excel-add-ins-javascript-programming-overview

[43] *Getting Started with VBA in Office 2010* [online]. 2016 [cit. 2016-04-05]. Dostupné z: https://msdn.microsoft.com/en-us/library/office/ee814735(v=office.14).aspx

[44] *VBA and Office Solutions in Visual Studio Compared* [online]. [cit. 2016-04-05]. Dostupné z: https://msdn.microsoft.com/en-us/library/ss11825b.aspx

[45] *Visual Studio Tools for Office (VSTO) - Craig Bailey* [online]. 2010 [cit. 2016-04-14]. Dostupné z: http://www.craigbailey.net/visual-studio-tools-for-office-vsto/

[46] *Create VSTO Add-ins for Office by using Visual Studio* [online]. [cit. 2016-04-14]. Dostupné z: https://msdn.microsoft.com/en-us/library/jj620922.aspx

[47] *Common Language Runtime Interop Layer* [online]. 2002 [cit. 2016-04-14]. Dostupné z: https://msdn.microsoft.com/en-us/library/ee796792(v=cs.20).aspx

[48] *Creating a fast Excel add-in (C#, VB.NET). Reading and updating cells.* [online]. 2011 [cit. 2016-04-20]. Dostupné z: https://www.add-in-express.com/creating-addins-blog/2011/09/29/excel-read-update-cells/

[49] *Compute Nodes - IT4I Docs* [online]. [cit. 2016-04-20]. Dostupné z: https://docs.it4i.cz/salomon/compute-nodes

[50] *C++ : mt19937 Example | Guy Rutenberg* [online]. 2014 [cit. 2016-04-20]. Dostupné z: https://www.guyrutenberg.com/2014/05/03/c-mt19937-example

[51] *Probability-distribution-image* [online]. [cit. 2016-04-22]. Dostupné z: http://www.strobertchs.com/teachers/ekotulski/148F834F-0119EDCC.51/Probability-distribution-image0.gif

[52] *Is HDF5 thread-safe?* [online]. 2016 [cit. 2016-04-22]. Dostupné z: https://www.hdfgroup.org/hdf5-quest.html#tsafe

[53] *IIS Worker Process* [online]. 2010 [cit. 2016-04-22]. Dostupné z: https://technet.microsoft.com/en-us/library/cc735084(v=ws.10).aspx

A Jednotlivé stupně otevřenosti OpenDat

1. stupeň – jedná se o obsah, který je dostupný na internetu pod otevřenou licencí, je snadné obsah vytisknout, přečíst, uložit, sdílet a popřípadě změnit. Obsah je snadné publikovat. Může být problematické data použít pro další strojové zpracování. Příkladem je formát PDF.

2. stupeň – obsah je dostupný jako strukturovaná data (uzavřený formát), je snadné obsah vytisknout, přečíst, uložit, sdílet a případně změnit. Obsah je nenáročné publikovat. Je snadné přímo zpracovávat data proprietárním softwarem pro agregaci, provádět výpočty, vizualizaci. Můžeme data exportovat do jiného (strukturovaného) formátu. Příkladem je Excel.

3. stupeň – platí totéž jako u 2. stupně, navíc data jsou dostupná pomocí neuzavřeného formátu. Můžeme snadno manipulovat s daty bez potřeby vlastnit jakýkoliv software, avšak může být potřeba nějakého konverzního nástroje nebo doplňku pro export dat do formátu uzavřeného. Příkladem je CSV.

4. stupeň – platí stejně jako ve 3. stupni, používá se URI (Unified Resource Identifier) k označení objektů. Můžeme odkazovat na data z jakéhokoliv místa (Web nebo lokálně). Můžeme znovu použít části dat, lze využívat knihovny a nástroje. Data lze bezpečně kombinovat s jinými daty na 4. a 5. stupni. Příkladem je RDF (Resource Description Framework) bez propojení, SOAP, JSON.

5. stupeň – platí rovněž jako ve 4. stupni, data jsou propojená pomocí odkazů na jiná data. Lze objevit více (souvisejících) dat při odběru. Můžeme se přímo dozvědět o datovém schématu. Data jsou dohledatelná. Související datové zdroje musí být také k dispozici minimálně na stupni 4 hvězdičky. Příkladem je RDF (Resource Description Framework) s propojením.

B Seznam zemí pokrytých komerčními službami na území Evropské unie

Tabulka 10: Přehledová tabulka pokrytí EU komerčními službami - část 1 [32]

Země	**Komerční služby**
Česká Republika	CE-Traffic: TMC, distri-bution channel: GSM/UMTS, Internet INRIX (TPEG/GSM basedservice) TomTom Here
Rakousko	INRIX TomTom HERE
Belgie	Be-Mobile: XML, DAB, RDS TMC INRIX TomTom HERE
Bulharsko	Be-Mobile: XML TrafficNav RDS-TMC HERE
Dánsko	Mediamobile RDS-TMC INRIX TomTom Here
Finsko	Mediamobile RDS-TMC TomTom Here
Francie	Michelin Travel Partner RDS/TMC and TPEG (Connected devices) Mediamobile RDS-TMC INRIX TomTom Here
Chorvatsko	Here
Německo	Mediamobile DAB-TPEG INRIX TomTom Here (internet, RDS-TMC, DAB-TPEG)

Tabulka 11: Přehledová tabulka pokrytí EU komerčními službami - část 2 [32]

Maďarsko	Be-Mobile: XML TrafficNav RDS-TMC TomTom Here
Irsko	TrafficNav RDS-TMC INRIX TomTom Here
Itálie	Infoblu INRIX TomTom Here
Lucembursko	Be-Mobile: RDS TMC, XML INRIX TomTom Here
Malta	TomTom
Nizozemsko	Be-Mobile: DAB, RDS TMC, XML ANWB: RDS-TMC, internet Verkeers Informatie Dienst (VID): RDS-TMC; Internet INRIX TomTom Here
Norsko	Mediamobile RDS-TMC, TPEG TomTom Here
Polsko	Mediamobile RDS-TMC Service provider: CE-Traffic: TMC Distribution channel: GSM/UMTS, FM RDS, Internet INRIX TomTom Here
Portugalsko	Be-Mobile: RDS TMC, XML INRIX TomTom Here

Tabulka 12: Přehledová tabulka pokrytí EU komerčními službami - část 3 [32]

Rumunsko	Be-Mobile: RDS TMC, XML Here
Řecko	Be-Mobile: RDS TMC, XML TrafficNav: RDS TMC, XML TomTom Here
Slovensko	CE-Traffic: TMC, distribution channel: GSM/UMTS, FM RDS, Internet RDS TMC, XML Be-Mobile: XML TrafficNav RDS-MTC Here
Slovinsko	TrafficNav: RDS TMC, XML Be-Mobile: XML Here
Španělsko	INRIX TomTom Here
Švédsko	Mediamobile RDS-TMC, premium smartphones apps INRIX TomTom Here
Švýcarsko	Viasuisse/SRG (RDS-TMC) TomTom Here
Ukrajina	Here
Anglie (UK)	Trafficmaster TMC via FM RDS Trafficmaster TPEG via DAB Trafficmaster TMC via Internet TomTom Here INRIX (TPEG/GSM based service) INRIX TPEG service (over DAB), INRIX TMC service (over FM RDS)

C Popis HDF5 webové služby

Jednotlivé metody lze volat pomocí SOAP nebo REST API, ovšem doporučuji používat službu pomocí SOAP. Volání pomocí REST API nelze používat pro zápis do HDF5.

C.1 Modul – Read

SOAP (WSDL): http://server/Read.svc?wsdl

C.1.1 GetBlockByColumn

Přečte blok struktury po sloupcích
Volání pomocí URL:
http://server/Read.svc/Read/GetBlockByColumn/{nameOfFile}/{dataset}/{groupName}
/{lowX}/{upX}/{lowY}/{upY}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)
lowX - od kterého sloupce včetně chci číst
upX - do kterého sloupce včetně chci číst
lowY - od kterého řádku včetně chci číst
upY - do kterého řádku včetně chci číst

Vrátí: HDFObject s naplněným příslušným polem
Příklad:
http://server/Read.svc/Read/GetBlockByColumn/testFile/Dataset/_/0/0/0/1
http://server/Read.svc/Read/GetBlockByColumn/testFile.h5/Dataset/_GroupA_GroupB/0
/0/0/1

C.1.2 Objekt HDFObject

Obsahuje:
traffic - pole objektů TrafficInfoClass
weather - pole objektů WeatherInfoClass

C.1.3 Objekt TrafficInfoClass

Obsahuje:
CREATED_TIMESTAMP - Čas publikování dat

DE - název úseku
PC - TMC kód
LI - unikátní řetězec pro úsek
PBT - poslední čas aktualizace
CF_FF - rychlost FreeFlow
PF_UT - doba, po kterou je pravdivá predikce
CF_SU - aktuální rychlost neomezena rychlostním limitem
SN - indikace, kdy se data naposledy změnila
CF_JF - úroveň zácpy
CF_TY - typ lokace
TABLE_ID - TMC tabulka
CF_SP - aktuální rychlost omezena rychlostním limitem
PF_SP - predikována rychlost
LE - kilometr měřeného úseku
EBU_COUNTRY_CODE - kód země TMC tabulky
CF_CN - věrohodnost dat v procentech
SHP_FC - počet křivek cesty

C.1.4 Objekt WeatherInfoClass

Obsahuje:
timestamp - časové razítko
name - název měřící stanice
degCel - stupně v Celsiích
degFah - stupně ve Fahrenheitech
lon - zeměpisná délka
lat - zeměpisná šířka

C.1.5 GetBlockByRow

Přečte blok struktury po řádcích
Volání pomocí URL:
http://server/Read.svc/Read/GetBlockByRow/{nameOfFile}/{dataset}/{groupName}
/{lowX}/{upX}/{lowY}/{upY}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP

fungují oba znaky)
lowX - od kterého sloupce včetně chci číst
upX - do kterého sloupce včetně chci číst
lowY - od kterého řádku včetně chci číst
upY - do kterého řádku včetně chci číst

Vrátí: HDFObject s naplněným příslušným polem
Příklad:
http://server/Read.svc/Read/GetBlockByRow/testFile/Dataset/_/0/0/0/1

C.1.6 GetBlockByRowByParameter

Přečte blok daného parametru struktury po řádcích (pro strukturu Traffic bude parametr 0, vrátí *timestamp*, parametr 1 vrátí název úseku apod.)
Volání pomocí URL:
http://server/Read.svc/Read/GetBlockByRowByParameter/{nameOfFile}/{dataset}
/{groupName}/{_parameterNumber}/{lowX}/{upX}/{lowY}/{upY}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)
parameterNumber - pořadové číslo parametru ve struktuře
lowX - od kterého sloupce včetně chci číst
upX - do kterého sloupce včetně chci číst
lowY - od kterého řádku včetně chci číst
upY - do kterého řádku včetně chci číst

Vrátí: Pole řetězců
Příklad:
http://server/Read.svc/Read/GetBlockByRowByParameter/testFile/Dataset/_/1/0/0/0/1
http://server/Read.svc/Read/GetBlockByRowByParameter/testFile.h5/Dataset/_/1/0/0/0/1

C.1.7 GetBlockByColumnByParameter

Přečte blok daného parametru struktury po sloupcích (pro strukturu Traffic bude parametr 0, vrátí timestamp, parametr 1 vrátí název úseku apod.)
Volání pomocí URL:
http://server/Read.svc/Read/GetBlockByColumnByParameter/{nameOfFile}/{dataset}
/{groupName}/{_parameterNumber}/{lowX}/{upX}/{lowY}/{upY}

Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)
parameterNumber - pořadové číslo parametru ve struktuře
lowX - od kterého sloupce včetně chci číst
upX - do kterého sloupce včetně chci číst
lowY - od kterého řádku včetně chci číst
upY - do kterého řádku včetně chci číst

Vrátí: Pole řetězců
Příklad:
http://server/Read.svc/Read/GetBlockByColumnByParameter/testFile/Dataset/_
/1/0/0/0/1
http://server/Read.svc/Read/GetBlockByColumnByParameter/testFile.h5/Dataset/_
/1/0/0/0/1
(pro název cesty)

C.1.8 GetPoints

Metoda přečte body struktury o zadaných souřadnicích
Volání pomocí URL:
Nelze volat pomocí URL!
Parametry:
data - objekt HDFPoints

Vrátí: HDFObject s naplněným příslušným polem

C.1.9 Objekt HDFPoints

Obsahuje:
nameOfFile
dataset
groupName
points - pole objektů Point

C.1.10 Objekt Point

Obsahuje:
col - souřadnice sloupce
row - souřadnice řádku

C.1.11 GetPointsByParameter

Metoda přečte body daného parametru struktury o zadaných souřadnicích
Volání pomocí URL:
Nelze volat pomocí URL!
Parametry:
data - objekt HDFPointsParameter

Vrátí: Pole řetězců

C.1.12 Objekt HDFPointsParameter (dědí z objektu HDFPoints)

Obsahuje:
nameOfFile
dataset
groupName
points - pole objektů Point
parameter – číslo parametru

C.1.13 FindRow

Metoda vrátí číslo řádku, na základě zadané časové informace
Volání pomocí URL:
http://server/Read.svc/Read/FindRow/{nameOfFile}/{dataset}/{groupName}/{_searchDate}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)
searchDate - časové razítko ve formátu Unix Timestamp

Vrátí: Celočíselnou hodnotu (Integer)
Příklad:
http://server/Read.svc/Read/FindRow/testFile/Dataset/_/1460252485

C.1.14 GetFileDatasets

Metoda vrátí seznam datových souborů v souboru HDF5
Volání pomocí URL:
http://server/Read.svc/Read/GetFileDatasets/{nameOfFile}/{dataset}/{groupName}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)

Vrátí: Pole objektů HDFMetadata

C.1.15 Objekt HDFMetadata

Obsahuje:
name - název objektu (identifikátor)
typeOfObject - typ objektu

C.1.16 GetDatasetDimension

Metoda vrátí rozměry datového souboru
Volání pomocí URL:
http://server/Read.svc/Read/GetDatasetDimension/{nameOfFile}/{dataset}/{groupName}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)

Vrátí: Pole bezznaménkových datových typů long (unsigned long)

C.1.17 GetStructNumOfParameters

Metoda vrátí počet parametrů datové struktury
Volání pomocí URL:
http://server/Read.svc/Read/GetStructNumOfParameters/{nameOfFile}/{dataset}/{groupName}
Parametry:

nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)

Vrátí: Celočíselnou hodnotu (Integer)

C.1.18 GetStructNames

Metoda vrátí názvy všech atributů datové struktury
Volání pomocí URL:
http://server/Read.svc/Read/GetStructNames/{nameOfFile}/{dataset}/{groupName}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)

Vrátí: Řetězec

C.1.19 GetGroupItems

Metoda vrátí objekty obsažené v HDF5 skupině
Volání pomocí URL:
http://server/Read.svc/Read/GetGroupItems/{nameOfFile}/{groupName}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)

Vrátí: Pole objektů HDFMetadata

C.1.20 GetAttributeDimension

Metoda vrátí rozměry daného atributu
Volání pomocí URL:
http://server/Read.svc/Read/GetAttributeDimension/{nameOfFile}/{dataset}/{groupName}
/{atributeName}

Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)
atributeName - název atributu datasetu

Vrátí: Pole bezznaménkových datových typů long (unsigned long)

C.1.21 ReadAttribute

Metoda vrátí přečtený atribut
Volání pomocí URL:
http://server/Read.svc/Read/ReadAttribute/{nameOfFile}/{dataset}/{groupName}
/{atributeName}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu v HDF5
groupName - název skupiny v HDF5 (pro URL volání pro znak / se používá znak _ , pro SOAP fungují oba znaky)
atributeName - název atributu datasetu

Vrátí: HDFObject s naplněným příslušným polem

C.2 Modul – HDF

Modul HDF spravuje úložiště HDF souborů
SOAP (WSDL): http://server/HDF.svc?wsdl

C.2.1 GetDatastore

Získá pole metadat uložených souborů
Volání pomocí URL:
http://server/HDF.svc/HDF/GetDatastore
Vrátí: Pole objektů HDFStore

C.2.2 Objekt HDFStore

Obsahuje:
id - identifikátor
fileName - název souboru
createTime - čas vytvoření souboru
lastWriteTime - poslední čas zápisu do souboru

C.3 Modul – Create

SOAP (WSDL): http://server/Create.svc?wsdl

C.3.1 Create

Vytvoří nový HDF soubor, skupinu či dataset
Volání pomocí URL:
http://server/Create.svc/create/{nameOfFile}/{datasetName}/{groupName}/{_isCompound}
/{_createNew}/{_compressionType}/{_compressionLevel}
/{_structType}/{_chunkX}/{_chunkY}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
datasetName - název datasetu
groupName - název skupiny
_isCompound - (true) zda se jedná o strukturu jako komplexní datový typ v HDF5 nebo (false) struktura je rozdělena do jednotlivých datasetů
_createNew - (true) vytváří nový HDF soubor, (false) přidává do HDF Groupy nebo Datasety
_compressionType – 0 = žádná komprese, 1=gzip
_compressionLevel - úroveň komprese v intervalu 1-9 (1 nejlepší rychlost, nejhorší kompresní poměr)
_structType - zvolení datové struktury volání dle čísla nebo názvu struktury (1,2 nebo Traffic, Weather)
_chunkX - velikost chunku pro osu X (minimální velikost chunku je 100)
_chunkY - velikost chunku pro osu Y (minimální velikost chunku je 100)

Vrátí: Boolean (true=soubor byl správně vytvořen)
Příklad:
http://server/Create.svc/Create/create/testFile.h5/Dataset/_/truetrue/1/5/1/100/100
http://server/Create.svc/Create/create/testFile/Dataset/_/truetrue/0/0/Traffic/100/100

C.3.2 CreateGroups

Vytvoří novou skupinu v HDF souboru
Volání pomocí URL:
http://server/Create.svc/create/{nameOfFile}/{groupName}/{_createNew}
Parametry:
nameOfFile - název souboru (s příponou i bez přípony)
groupName - název skupiny
_createNew - (true) vytváří nový HDF soubor, (false) přidává do HDF Groupy nebo Datasety

Vrátí: Boolean (true=skupina byla správně vytvořena)

C.4 Modul – Write

SOAP (WSDL): http://server/Write.svc?wsdl

C.4.1 Append

Metoda zapíše data na začátek nového řádku za aktuálními daty
Volání pomocí URL:
Nelze volat pomocí URL!
Parametry:
data - objekt HDFWriter

Vrátí: Boolean (true=zápis se podařil)

C.4.2 Objekt HDFWriter

Obsahuje:
nameOfFile - název souboru (s příponou i bez přípony)
dataset - název datasetu
groupName - název skupiny
numOfTimeSeries - počet vkládaných časových řad
numOfRecords - počet vkládaných záznamů
traffic - pole objektů TrafficInfoClass
weather - pole objektů WeatherInfoClass

(pokud je definováno více různých objektů v HDFWriteru, pak se bere první pole objektů dle

pořadí, př. vložil jsem Traffic a Weather, při zápisu bude zvolen objekt Traffic – má nižší číslo ve výčtu)

C.4.3 WriteAtLocation

Metoda zapíše data na přesné místo (souřadnice) v datasetu
Volání pomocí URL:
Nelze volat pomocí URL!
Parametry:
data - objekt HDFWriter
column - souřadnice sloupce
row - souřadnice řádku

Vrátí: Boolean (true=zápis se podařil)

C.4.4 WriteAttribute

Metoda zapíše data do atributu v datasetu
Volání pomocí URL:
Nelze volat pomocí URL!
Parametry:
data - objekt HDFWriter
atributeName - název atributu

Vrátí: Boolean (true=zápis se podařil)

D Obsah přiloženého CD

Přiložené CD obsahuje následující adresáře:

- Diplomová práce - text diplomové práce ve formátu PDF
- HDFAdapterSolution - složka obsahující projekty
  - HDF_Wrapper_CLI - složka obsahující projekt adaptéru pro řízené `C++/CLI`
  - HDF_Wrapper_CLI_UTests - složka obsahující unit testy pro adaptér pro řízené `C++/CLI`
  - HDF_Wrapper_CPP - složka obsahující projekt adaptéru pro `C++`
  - HDF_Wrapper_CS - složka obsahující projekt adaptéru pro C#
  - HDF Excel AddIn - složka obsahující projekt doplňku pro Microsoft Excel
  - HDFCompareAdapters - složka obsahující projekt pro porovnání výkonu adaptérů `C++/CLI` a kombinaci adaptérů `C++` a C#
  - HDFPerformanceTester - složka obsahující projekt pro testování výkonu `C++` adaptéru
  - HDFService - složka obsahující projekt HDF5 webové služby
  - HDFWebClient - složka obsahující projekt klientské knihovny webové služby
- GeneratorSolution - složka obsahující projekt pro generování dat